Collaborative Innovation

strategy, technology, and social practice

ETH Habilitation

COLLABORATIVE INNOVATION

STRATEGY, TECHNOLOGY, AND SOCIAL PRACTICE

A habilitation submitted to
ETH Zurich for a Venia Legendi
in the Management of Innovation

by
Stefan Haefliger

Citizen of Lucerne
born on September 6, 1976

Habilitation Committee Members
Prof. Dr. Stefano Brusoni, ETH Zurich
Prof. Georg von Krogh PhD, ETH Zurich
Prof. Dr. Nikolaus Franke, WU Vienna
Prof. Charles Baden-Fuller PhD, Cass Business School

2012

To my fellow practitioners:

Edith Lilith, Georg, Dominik, Sebastian, Gauri, Martin, Sabina, Fotini, Heinz, Marion,
Pablo, Zeynep, Boris, Jeannette, Lise, Cristina, Robert, Peter, Daniella, Ele, Patricia,
Monorom, Simon, Zai, Efe, Silvan, Fredrik, Stefan, Philipp, Daved,
Alban, Evila, Nina, Renato, Hilde, Helena, Andreas

Acknowledgements

My gratitude goes first of all to my co-authors who have significantly contributed to the adventures in thinking and writing that lead to the discoveries that I can share here. They are: Georg von Krogh, Sebastian Spaeth, Peter Jäger, Martin Wallin, Simon Gächter, Patricia Wolf, Philipp Reichen, Dominique Foray, and Eric Monteiro. This thesis is the result of collaborations that span over more than a decade. It gathers eight publications in some of our field's finest outlets and I could not have achieved this level of quality by myself. In a line of work as invasive and permeating most boundaries in life as research it is a great privilege to know my co-authors as personal friends. Thus, my gratitude goes far beyond academic work to the shared practice that carves large and small discoveries out of the stream of thoughts and shared experiences in inner and outer worlds as we balance them every day.

Much the same is true for dear colleagues and co-authors I've had the great fortune of collaborating with on works including proposals, letters, chapters, and articles that could not be directly integrated into this thesis. I've received sustained support, invaluable learnings, and loving encouragement over the years. These dear friends are: Stefan Meisiek, Eric von Hippel, Cristina Rossi Lamastra, Matthias Stuermer, Carliss Baldwin, Marion Poetz, Massimo Colombo, Fotini Pachidou, Alessandro Rossi, Evila Piva, Daved Barry, Charles Baden-Fuller, Jiro Nonaka, Raghu Garud, Simon Grand, James Robins, Philipp Tuertscher, Markus Geipel, and all the colleagues here at ETH!

The research included here enjoyed the generous support of my employer over the last six years, ETH Zurich, and from the Swiss National Science Foundation through our two grants: 100014_125513 and 100012-101805.

Abstract

Collaborative innovation is growing rapidly, enabled by the internet and the possibilities it offers to share ideas, content, code, and opinions fast and cheaply across the globe. It is relevant for business and has caught the minds of management thinkers because of the dynamics and promises that distant collaboration and the reuse of knowledge entail. Our work, introduced here, has followed three trajectories over the last years that can be captured in shorthand as follows: we described phenomena in strategy, we followed artifacts (code), and we observed collective action (social practice). This collection of essays submitted as a habilitation in management science maps theses three trajectories of work that capture three essential characteristics of collaborative innovation: its strategic relevance, the technological agenda, and the communal work as the social practice. Each of these characteristics can be seen as a perspective that opens up fascinating research questions and an agenda for management science to be covered in future research.

TABLE OF CONTENTS

TABLES

Figures

COLLABORATIVE INNOVATION:

STRATEGY, TECHNOLOGY, AND SOCIAL PRACTICE

Stefan Haefliger

Chair of Strategic Management and Innovation
Department of Management, Technology, and Economics (MTEC)
ETH Zürich, Switzerland

January 2012

1. INTRODUCTION AND DEFINITION

Collaborative innovation is defined here as a development process of new and useful products and services across and outside firm boundaries. This definition helps to characterize a research agenda for organization and management science that takes seriously three central approaches to theory building and testing that deserve attention: strategy, technology, and social practice. They deserve particular theoretical attention due to the speed of technology development (mostly in information and communication technologies ICT) and recent developments in our field which point to a shift in theorizing towards assessing real life problems and action in the context of their inseparable social and material characteristics (Orlikowski and Scott, 2008). First, a brief explanation of the definition, then the three approaches, then the justification in terms of theory and relevance.

Defining collaborative innovation as a development process of new and useful products includes the study of user innovation because user innovations are also defined as new and useful independent of commercialization and market success (von Hippel, 1988). The definition focuses on an activity over time rather than on states and transactions. Collaborative innovation, here, refers to a creation and development process involving multiple actors and stakeholders inside and outside of firms who collaborate around a specific purpose of generating ideas, concepts, technologies, and solutions for business or for their own use. Out of focus moves the market for technology (Arora, Fosfuri and Gambardella, 2001) as well as the similar concept of open innovation (Chesbrough, 2003) because while trading licenses and patents enables innovation inside and outside of firms, the process of generating and developing ideas is frequently excluded or assumed in that literature.

Firm boundaries play a critical role in collaborative innovation as defined here. Development processes span across organizations and include individual actor outside of firms and in networks that include universities, communities, and partner firms (Allen, 1983; Mowery, 1983; von Hippel, 1988; Powell, Koput and Smith-Dorr, 1996; Prandelli and Sawhney, 2000; von Hippel and von Krogh, 2003). Collaborative innovation has been documented historically as collaborations between independent research organizations and manufacturers (Mowery, 1983) and the free exchange of knowledge (techniques, designs, know-how) among firms (Allen, 1983, von Hippel, 1987). Learning is known to occur in networks of firms (Powell et al., 1996) and individuals and firms engage in joint development activities in private-collective innovation (von Hippel and von Krogh, 2003). Firms contribute to technology development outside their boundaries over long periods of time by giving away technology and participating in development activities (Henkel, 2006; Stuermer, Spaeth and von Krogh, 2009).

This collection of works highlights three approaches to collaborative innovation that extend into an agenda for future research. They are strategy, technology, and social practice. Each approach had been tackled with a slightly different set of methodologies and outlook at theory and theory development. First, strategy is about enabling collaborative innovation from the perspective of management. It is about efficient and feasible ways of collaborating and learning from external users, peers, and consumers and how to organize in order to collaborate.

Second, technology enables collaboration independent of the actors who are in charge, deploying certain communication and information infrastructures in their communities or firms. Technology is an artifact with the ability to influence and profoundly change the way work is organized (Leonardi, 2007). Studying collaborative innovation by following the technology means identifying the knowledge that enables action, which is code, documentation, designs, text, and much more.

Third, collaborative innovation as a process refers to a social practice. The social practice turn in the field of organization and management science has sharpened our perception of everyday work in context (Schatzki, Knorr-Cetina and Savigny, 2001). Studying the social practices that constitute collaborative innovation means studying the working environment of individuals in communities and firms

who engage in discussions and exchanges with other individuals often located at a distance both physically and organizationally. Understanding why individuals engage and how they build productive, creative, sustaining working environments (institutions, routines, relationships) requires novel assumptions about motivation and incentives in organizations.

The first theoretical reason why we should study collaborative innovation is straightforward. ICT has changed the life and work environments of almost everyone in the developed and developing world. The existent theories that explain how members of organizations collaborate with individuals outside the organization usually, up to say 10 years ago, cannot cover the implications of wide-spread social software applications or even access to the Internet. This claim needs refinement which I will only touch upon lightly in this introduction and refer to the individual papers in this collection to carve out the space of their theoretical contribution. However, three warrants stand out: access, speed, and network characteristics of ICT.

The Internet counts over 2bn users as of March 2011[1]. Individuals with needs and preferences from all walks of life can access distributed sources of information and contribute their ideas, designs, programs, or support. Consider the project Open Source Ecology (OSE)[2]. Marcin Jakubowski founded the rapidly growing network of farmers, engineers and supporters in 2003 with the goal of prototyping the fifty basic industrial machines needed for a sustainable civilization with modern comforts in an easy, do-it-yourself fashion. Access to the Internet from remote corners of Earth enable farmers everywhere to build their own machines using the OSE blueprints and documentation material, such as video tutorials. Twenty years ago, the growth of Open Source software demonstrated that distributed software developers organized in online communities in order to download and build on each others' code (Stallman, 1999; Moody, 2001). Thus, explaining how a farmer in Missouri may work together with a farmer in Cambodia needs to take into account that both may have access to the same online resources.

Speed of ICT needs to be considered: consumer feedback may come back seconds after product release just as competing and collaborating groups of crackers race to publish the first illegal copy of a newly released game or piece of software (Rehn, 2001). A theory of collaborative innovation needs to build on the assumption that content can be shared across the globe instantaneously just as funds can be transferred within seconds. In addition to access and speed, ICT offer network characteristics that allow individuals to connect to each other in selective and dynamic ways giving rise to what has been described as fluidity (Faraj, Jarvenpaa and Majchrzak, 2011). Faraj and colleagues (2011) start and call for theory on the organization of dynamic resource flows in knowledge collaboration and characterize tensions and generative responses in the organization of collaborative work in online communities. The network characteristics of ICT enable individuals and firms to enter and exit networks by contributing and withdrawing resources such as time and passion. Organization and management theory stands at the outset of understanding what ICT affords in terms of collaborative innovation.

The second theoretical warrant for studying collaborative innovation has been supplied by a fruitful and rapidly evolving perspective in organization and management theory, broadly known as the practice turn. Studying innovation means studying processes of generation and experimental and ambiguous action towards a goal that is only partly understood and uncertain from the start. We cannot know what we do not know yet. This truism leads to endorsing a perspective in theorizing that can deal with complexity, evolving structures and rules, dynamic interactions, and individuals who enact organizational reality facing ever changing contexts and external forces (Ciborra, 1996; Feldman and Orlikowski, 2011).

[1] According to Internet World Stats: http://www.internetworldstats.com/stats.htm

[2] For more information turn to: http://opensourceecology.org/about.php

Collaborative innovation generates artifacts and is in turn influenced by the artifacts produced along the way. On a fundamental, theoretical level, we follow Leonardi (2011) when suggesting that new technologies and new routines are brought about by both human and material agencies. Thus collaborative innovation can be described in terms of the social interactions as well as in terms of the material progress and contingencies that co-determine the path innovation takes. In reality, these descriptions are inseparable and refer to an entangled whole (Barad, 2003; Orlikowski, 2010) which requires careful theorizing in terms of attention to context, timing, and the materiality of collaborative innovation.

Lastly, collaborative innovation is practically relevant. Customers of mass customized products are willing to pay more when able to design the product themselves (Franke, Schreier and Kaiser, 2010). Open source software developed collaboratively over the Internet is used to run 75% of websites globally and the world's largest stock exchanges run Linux systems when attempting to reach new records in trading speed (King 2010; Vaughn-Nichols 2009). Pharmaceutical companies form research clubs with competitors and research institutions that will release their discoveries into the public domain while the companies get to influence the research agenda by nominating proteins of potential value to their products (Alexy, Criscuolo and Salter, 2009).

Collaborative innovation	Contributions	Publications
Strategy How to enable collaborative strategy?	Skill development and experimentation with fellow users for industry entry by user entrepreneurs Initiating private-collective innovation through structures and licenses that facilitate knowledge sharing Framework for understanding the application of social software in strategy	Haefliger, Jäger, von Krogh, 2010, ***Research Policy*** Gächter, von Krogh, Haefliger, 2010, ***Research Policy*** Haefliger, Monteiro, Foray, von Krogh, 2011, ***Long Range Planning***
Technology What technologies are developed and reused?	Code reuse in open source software identifying the logic of reuse behavior among developers Designing for reuse and joint development by drafting an open approach to design science in IS Reuse and forking as mechanisms of distributed systems integration in open source software development	Haefliger, von Krogh, Spaeth, 2008, ***Management Science*** von Krogh and Haefliger, 2010, ***Journal of Strategic Information Systems*** Haefliger, 2012, ***Working paper***
Social practice Who collaborates and why?	Motivation of open source software developers; conceptual work with new assumptions about individual motivation Modding behavior and interaction in a large community of Machinima fans and consumers Participation in intra-firm communities of practice in the automotive industry	von Krogh, Haefliger, Spaeth, Wallin, 2012, ***MIS Quarterly*** Haefliger, Reichen, Jaeger, von Krogh, 2009, ***Lecture Notes in Computer Science*** Wolf, Spaeth, Haefliger, 2011, ***Journal of Knowledge Management***

Table 1: Overview of thesis chapters with published articles

This habilitation thesis is structured according to the logic laid out in Table 1. Each row represents a section: strategy, technology, and social practice. By submitting as the thesis a collection of nine articles (eight of which published) each section briefly discusses the articles and the research issues emerging from each work. The concluding section ties back to this introduction by formulating research questions that point beyond the articles and working papers to inspire a broader research agenda on collaborative innovation.

2. STRATEGY

> *"Well, we actually, in the first step so we made fun of Sony Vista and so, no not Sony, Microsoft Vista and it's and we realized, oh shoot we're kind of, we're harsh about it and it's one of their clients. So we said, "Oh! Okay sorry, sorry guys."*
>
> *Interview with Frank Dellario of The Ill Clan, Brooklyn NY, March 27, 2007*

Strategy in collaborative innovation is a fragile affair. The quote above hints at three points about collaborative innovation and strategy: the creative potential of users, their disruptive potential to hurting business, and the fragility of interaction between users and firms when crafting strategy collaboratively. This first section on strategy explains crucial aspects of collaborative innovation through the difficulties of mutually aligning the interests of creative users with business and vice versa, the strategies to build a community of customers and draw on a community of peers, the conditions and fragility of initiating mutual sharing of knowledge, and the particular challenges when using social software as a means to connect to users and peers outside the firm.

The first section on strategy contains three essays contextualized here. First, the study of Machinima builds on 7 in-depth case studies that revealed a pattern of market entry by user entrepreneurs after and while collaborating with peers and learning how to create films using video games. Second, the case of the Machinima companies and the study of open source software development revealed the critical strategic issue of licenses and incentives that build a fundament for collaborative innovation. The experimental study of initiating private-collective innovation demonstrates the fragility of knowledge sharing and paves the way for the third, broader essay in this section. Mutual sharing of knowledge, collaboration that spans organizational boundaries, implies agreement about the benefits of engaging in collaboration. However, underlying successful cases of collaborative innovation between firms and communities or individual users are ICT tools that enable value creation and appropriation, their acceptance and use in firms, as well as leadership issues. The third essay elaborates on a research agenda for strategy and social software and builds a framework to help managers understand strategic issues when deploying social software in companies.

Frank Dellario is one of the fathers of Machinima. Co-founding the Ill Clan in 1997 with Paul Marino, Matt Dominianni, and Manu Smith in Brooklyn NY, the clan produced a series of videos that helped defined what the genre would become: shooting film inside video games meant taking a video game and manipulating its visual appearance and subtracting the interactive nature of the game that it was designed for. Frank and Paul re-skinned entire sequences of the game Quake in order to turn a first person shooter game, released by ID Software, into a cooking show. Re-skinning refers to the re-design of the artwork in the game for characters and scenery to no longer look like soldiers and space stations but, in this case, cooks and kitchens. Frank and Paul then went on stage. They used the game to perform a talk show in front of live audiences by "playing the game" while speaking over it. The game Quake, however, no longer looked like the monster hunting playground set on an alien planet but rather like a kitchen. The comical effect was overwhelming. They decided to record sessions and publish them online. The Ill Clan's legendary show "Common Sense Cooking with Carl the Cook" was born and with it inspiration for numerous other Machinimators who used video games to shoot film using various techniques and strategies for reaching an audience and turning their ventures into businesses.

Dellario's statement quoted above also shows that users may display problematic behavior for companies. The history of Machinima is fraught with potential and actual cases of copyright infringements that are characteristic of the struggle between users and copyright owners when using new technologies that allow re-using content and the practice of "rip-mix-burn" (Lee, 2008; Depoorter, 2009). In addition, defamatory statements are frequent and part of political and humorous expressions online. The legal uncertainty and the ease of reusing content may lead to a culture of suspicion between content owners (frequently firms, video game publishers in the case of Machinima) and creative users. Collaborative innovation, however, requires building on each others' works and respecting each others' positions as ideally business enabling. The company Rooster Teeth Productions, introduced in the next section, successfully established a relationship with Bungie, a former subsidiary of Microsoft, to use their video game sequels Halo to produce their Machinima series Red vs. Blue. Their collaboration has been ongoing for over nine years due in part to Bungie's conviction that the publication of the film series Red vs. Blue was actually positive for the sale of the video game Halo.

The achievement of agreement between users and firms for collaborative innovation should not be underestimated. Our cases studies in Machinima document widespread anxiety among both users and game publishers about the potential harm that could come from using video games in ways unintended by the designers. Machinimators, in turn, fear lawyers blocking the publication of films after production. An example of the first case was Electronic Arts (EA), a large game publisher, taking legal action to block the publication of a user-created modification of the game Battlefield to take place in Iraq. An example of the second case is a Machinimator creating an entire film with the ambition to submit it to a film festival and being denied taking part in the competition due to missing legal documents that could certify that the film did not contain copyrighted material not in the possession of the Machinimator. In this case, the video game publisher denied the Machinimator the right to use their video game for producing a film after the film had been produced. This unfortunate case illustrates painfully how collaborative innovation should not take place: without collaboration.

Agreement on the terms of reusing content is instrumental for collaborative innovation. Whether content is freely revealed by one party (Harhoff, Henkel and von Hippel, 2003) or contractually shared between two parties (Zhang and Baden-Fuller, 2010; Grant and Baden-Fuller, 2004), both strategies enable the process of collaborative innovation. The mostly successful cases of collaborative innovation in the first essay should not blind our perception of the difficulties of reaching agreement on a fundamental level. Knowledge sharing is most beneficial when mutual and this is where the second essay takes hold.

The notion of private-collective innovation (von Hippel and von Krogh, 2003) has profoundly influenced how organization theory can make sense of collaborative innovation. The study of private investments in innovations in firms (Arrow, 1984; Demsetz, 1967) stood apart from the study of investments in innovations outside of firms, in academia and collective action using public funds (Stephan, 1996; Olson, 1965). In the former, private investors receive the gains from selling the innovation; in the latter the rights to commercialize the innovation are relinquished in the interest of creating a public good. The hybrid between the two models describes the situation of private investors relinquishing control over their innovations and turning them into public goods. Von Hippel and von Krogh (2003) argued that the hybrid model, that they called the model of private-collective innovation, would work when private investors perceive greater benefit of revealing their innovations than from keeping it secret. Private-collective innovation could be observed in open source software development and in many cases of user innovation. The important empirical question was: when and why would the net benefit of revealing outweigh the potential benefits of exclusive appropriation of rents from the innovation?

One answer to this question involves the process of collaborative innovation. From studies of open source software developers it became clear that developers valued the involvement in collaborative

innovation in addition to the benefits that the actual, final product offered (Lakhani and Wolf, 2005; Spaeth, Haefliger, von Krogh and Renzl, 2008). These benefits include learning from peers, the ability to influence the technological agenda of the group of developers, enjoyment, feedback on own work, and so on. However, while these findings might explain why developers keep contributing to an ongoing development process it is harder to understand how private-collective innovation could begin. It is well known that the reasons for joining a collective action differ from the reasons for continuing because learning how collaboration works and identifying with the collective endeavor change motivations and the perception of costs and benefits (Elster, 1986; Shah, 2006).

Given the difficulty of observing the choice not to join collaborative innovation we faced a methodological challenge. In open source software development silent observers and users are called lurkers (Nonneke and Preece, 2000). Lurkers remain by definition invisible and if they were to respond to any form of request they would no longer be lurkers. A recent study of peripheral contributors to collaborative innovation by Setia, Rajagopalan, Sambamurthy and Calantone (2011) found that peripheral contributors enhanced and popularized a product, open source software in their case, by contributing bug reports and word-of-mouth promotion. Given the impossibility of studying the decision not to share knowledge in the field we modeled the incentive structure of this private decision in order to observe the behavioral consequences of the individual, private decision. The laboratory study revealed sharing in cases where no sharing would have been expected and fragility in terms of opportunity costs.

The third essay on strategy formulates a framework that both cautions against simplistic visions of collaborative innovation and calls for more research on the tools employed to enable collaborative innovation. Both issues are of critical strategic relevance. Under the topic 'Social Software and Strategy' the essay explores a specific growing family of ICT tools: social software is receiving considerable attention from managers and investors due to, for example, the considerable successes of initial public offerings of companies based on services offered within their own or others' social software networks, such as LinkedIn. Zynga, a publisher of online games that use Facebook as the equivalent to a gaming console, has raised enormous expectations only to see their growth rate slow down again and established game producers such as EA catching up[3].

First, the risk of simplistic metaphors among strategists may cloud their perception of the real challenges when introducing social software tools within and across their firm boundaries. In online communities, users frequently rely on social software to organize their work and "develop a life of their own" (Wiertz and de Ruyter, 2007: 390) deciding which relationships to build and when to judge company interventions as obtrusive. Partners in collaborative innovation may emerge among consumers, users, suppliers or even competitors. When consumers become co-developers in product design (Fuchs and Schreier, 2011) their power over strategic decisions may increase for good or worse. The metaphors of "harnessing" the creativity of the crowd and so on may blind out the potentially disruptive impact empowered users may exert on the firm.

The potential disruption of organizational routines and structures stems from the fact that the perspective from inside the firm may not necessarily converge with a perspective from outside the firm. While differing perspectives are a business reality, collaborative innovation may float them unexpectedly. The reason for this is that ICT has the capacity to change the way users seek and access and distribute information (Kling and Scacchi, 1982; Markus, 1983; Leonardi, 2007; 2008). In addition, when building a community designed for collaborative innovation membership is unpredictable and the members may be unknown to the sponsor or gatekeeper of the community (West and O'Mahony, 2005).

[3] According to the Wall Street Journal, Sept 21, 2011:
http://online.wsj.com/article/SB10001424053111903703604576585570928213398.html

Second, a strategy for collaborative innovation calls for detailed insights on how to use tools such as social software. A new study by Jarvenpaa and Lang (2011) calls for attention to boundary management when a firm sponsors a community or builds a platform for collaborative innovation. Openness and control represent a trade-off closely linked to the boundaries that online communities share with the firm. Issues such as power, identity, and competence should be taken into consideration at the same time when making boundary decisions. The authors highlight the notion of "generative capacity" which enables knowledge collaboration in online communities and responses to tensions inherent in organizations characterized by unpredictable membership fluctuation and dynamic rather than contractually secured resource commitments (see also Faraj et al., 2011).

UNDER THE RADAR:

INDUSTRY ENTRY BY USER ENTREPRENEURS

Stefan Haefliger
Peter Jäger
Georg von Krogh

Chair of Strategic Management and Innovation
Department of Management, Technology, and Economics (MTEC)
ETH Zürich, Switzerland

Published in Research Policy vol. 39 issue 9 2010 pages 1198-1213

We inductively develop a model of the commercialization process for new products or services user entrepreneurs undertake when entering an industry while drawing on proprietary technology developed in another industry. Extending the growing field of user entrepreneurship, we identify a two-phase approach to industry entry by user entrepreneurs who start "under the radar" of incumbent firms, gain experience, attract a first potential customer base, and then, in a second phase, engage in commercialization. During this process, a community of fellow users is of major importance for the entrepreneur, serving as a knowledge pool for skills development and experimentation with different commercialization paths. We study a nascent group of firms founded by users of video games who became entrepreneurs on entering the animation industry by producing Machinima, a new film genre characterized by shooting film in video games. We explain how user entrepreneurs gain access to complementary assets (video games) for their new use (shooting film), how they deal with intellectual property issues when using other firms' assets, and how user entrepreneurs combine domain knowledge about film production with their experience in video games and the art of Machinima. Our propositions hold implications for management and policy.

INTRODUCTION

Entrepreneurship, conventionally understood, is a process where opportunity recognition precedes prototype development (Venkataraman, 1997; Shane and Venkataraman, 2000). In the case of user entrepreneurship, however, this process is reversed: users first develop prototypes and, while using and gaining experience with the new design, recognize a potential for commercialization of their product[4] or service (Shah and Tripsas, 2007). Thus, user entrepreneurship can be seen as taking a position within a standing debate in entrepreneurship theory about the discovery or creation of opportunities (Alvarez and Barney, 2005, 2007; Sarasvathy, 2001, 2004) by describing entrepreneurial opportunities as created in use. Generally showing traits of user innovators (von Hippel, 1988), user entrepreneurs derive their designs from existing products or technologies. If commercial value is created in a different industry from that in which the original product was located, we should expect new challenges connected to technology diffusion and knowledge recombination (Geroski, 2000). The aim of this paper is to fill a gap in the fast growing literature on user entrepreneurship, by theorizing about the process of user entrepreneurship as users move from one industry to another in order to commercialize. User entrepreneurs face low opportunity costs and exhibit a high willingness to experiment and high potential to explore commercial opportunities by entering existing markets or creating new ones, especially when the target markets are turbulent and demand is uncertain (Shah and Tripsas, 2007). Usually embedded in a community of users with similar needs, user entrepreneurs operate under favorable conditions: the community plays a vital role in diffusing new designs while user entrepreneurs are granted early access to feedback and information relevant to commercialization prior to firm foundation (Shah and Tripsas, 2007).

Examples of user entrepreneurship have been studied in the fields of sporting equipment (Baldwin et al., 2006; Luthje et al., 2005; Franke and Shah, 2003) and juvenile products (Shah and Tripsas, 2007). Both fields present cases where commercialization occurred in the industry in which the original user activity took place. In the case of juvenile products, dissatisfied parents innovated more sophisticated designs, like a stroller that can be used while jogging, which they eventually commercialized after other parents became aware of its superior performance. In the case of rodeo kayaking, users redesigned kayaks to better perform under extremely challenging conditions. In either case, user entrepreneurs developed new concepts on the back of existing products. They demonstrated and promoted their designs in competitions. In scanning for products that satisfied their needs prior to developing their own, user entrepreneurs gained familiarity with the general market structure and the competitive environment of their industry.

Products and services are fungible in their application. They can be modified, developed, or reinterpreted to be used as a tool or basis for new products in another context (Faulkner and Runde, 2009; von Hippel, 1988). User entrepreneurs may shift the use activity from one industry to another or, to put it differently, diffuse technology across industries through different use (cf. Rogers, 1962). Research has so far neglected this important aspect: user entrepreneurs who develop products based on assets from one industry, which they apply as complementary assets in the industry where they commercialize. Based on a study of the Machinima phenomenon, a new animation genre, and a sample of firms started by users, we develop a new process model of user entrepreneurship across industries. The firms in our sample apply video games as production technology for animation and commercialize their films via online distribution or DVD sales in the animation production industry, a subdivision of the motion picture industry. In the 1990s, user innovators started to record their game play and introduced recording technology to video games, transforming their use (Faulkner and Runde, 2009). With the publication of the first Machinima films, shot in video games, the technology became amenable to story telling that extended beyond the story elements contained in the game. User entrepreneurs publish animated

[4] To reduce complexity throughout the paper, we use the term "product" to refer to products, technologies, or processes.

shorts[5] using video games as production technology and create opportunities to commercialize from their experience with Machinima. Figure 1 shows the process of user entrepreneurship in two phases as we observe and develop it in this paper, where an initial user innovation in industry A enables user entrepreneurs to enter industry B, in a first phase, under the radar of incumbents and, in a second phase, to commercialize animated shorts based on the new use of a technology from industry A. For example Rooster Teeth Productions, a Machinima firm in our sample, made use of the game Halo, produced by Microsoft/Bungie, to create a Machinima franchise[6] called Red vs. Blue under which they publish episodes online. Based on their experience and audience gained, they subsequently sold DVDs, sponsorship access, and merchandising.

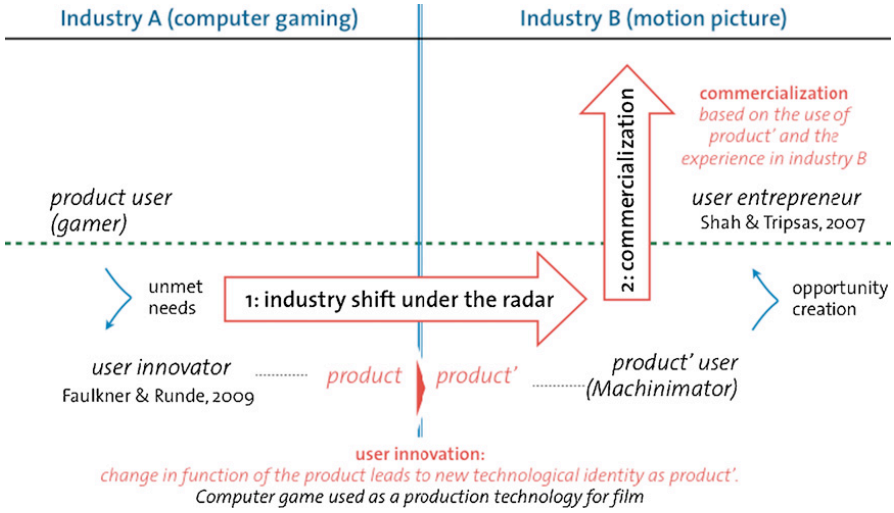


Figure 1: Process of user entrepreneurship in two phases

Products, goods, or services are usually sold or licensed under conditions that restrict their use or application, such as enduser license agreements (EULAs). User entrepreneurs drawing on existing products experience use restrictions as obstacles to commercialization. Relying on third party assets complicates commercialization; intellectual property rights attached to either modified products or assets remain with the original producer and create legal uncertainty for the user entrepreneur (Depoorter, 2009). Companies holding the IP of products affected by such user activity may show tolerance toward the application of their assets (Harhoff et al., 2003); otherwise, permission may be granted through a variety of means, such as research exemption in the field of science (O'Rourke, 2000; Strandburg, 2008), fair use under US copyright law, or the growing practice of (informal) unauthorized use of copyright material (Lee, 2008).

While firms in the industry where the original user activity was located (industry A) may apply and enforce patents or copyright to secure profits in their home market, they might adopt a more lenient position toward others' exploitation of these assets in industries where they do not compete (industry B), especially given the high cost and effort of monitoring and enforcing intellectual property rights (Liebeskind, 1996). Hence, firms may be willing to share selectively and tolerate the application of specific assets that are core to their "home" market but complementary in other markets, and so provide user entrepreneurs with a foundation to commercialize.

The knowledge needed to commercialize in another industry's markets may extend beyond the knowledge acquired in use and, possibly, beyond the experience available through the community of users (Baldwin et al., 2006). Domain knowledge important to commercialization includes market-relevant education and experience, production techniques, work flows and processes, insights about genres and market demand, and industry-specific marketing knowledge. We study commercialization by user entrepreneurs outside the industry where the products they use originated. This offers new insights into tolerance toward application of IP, opportunity creation by user entrepreneurs in new industries with fragmented markets, and community support across industries. The particular mix of competition, versatility of commercialization-relevant assets across industries, intellectual property rights, and the role of knowledge acquisition raises an impor-

[5] Animated shorts are the predominant products while (feature) films appeared rarely, especially early on. Throughout this paper, we use the terms film, animation, and shorts interchangeably.

[6] Walt Disney pioneered this marketing concept evoking various sources of revenue based on their characters (Wasko et al., 1993; Yoon and Malecki, 2009).

tant question that we address in this paper: Under what conditions do users, who apply (selectively shared proprietary) assets from one industry, commercialize in markets of another industry (thus becoming user entrepreneurs)?

We examine this question in the motion picture industry, which represents an ideal context to study user entrepreneurship, considering Shah and Tripsas' (2007) proposition that markets with high turbulence and demand uncertainty favor user entrepreneurship: besides a huge mainstream market for theatrical film production, several niche markets within the industry are in a state of revolt due to technological advances. Today, major studios show little interest in providing content for the Internet even though there is excess demand for it and customers are willing to pay. Industry analysts have commented that the low profitability of the Internet market, compared to their mainstream business, has meant that major studios have missed the opportunity to develop sustainable business models to serve online customers (Papies and Clemet, 2008). This neglect may prove critical as the Internet is gaining in importance as an outlet for media content (Scott, 2004; Yoon and Malecki, 2009). Furthermore, it is still unknown whether theatrical and non-theatrical outlets substitute or complement each other (Eliashberg et al., 2006). "The development of new delivery systems will in principle open up the market to more effective contestation by smaller independent film production and distribution companies (cf. Leyshon, 2001). Thus, the eventual attainment of film distribution by means of the Internet will no doubt give rise to a great increase in the amount of cinematic material available to consumers, thereby widening the market and almost certainly making inroads on blockbuster audiences" (Scott, 2004: 58).

The motion picture industry can expect to see new entrants from the video game industry since both industries show similar characteristics and boundaries are blurring (Calantone et al., 2010; Eliashberg et al., 2006; Yoon and Malecki, 2009). The animation industry is probably the closest link to the video game industry, in that both share substantial talents (Aoyama and Izushi, 2003; Izushi and Aoyama, 2006). With the increasing digitalization of the value chain, entry barriers to the animation industry are lowering, enabling everyone with a personal computer to participate (Eliashberg et al., 2006). Internet and animation production technologies overlap (Britton et al., 2009). Producing animated shorts is considered a point of entry for small studios that might later attract a growing audience. Being able to shift among markets means that these animation studios can eventually move into feature films, as AKOM demonstrated with The Simpsons (Yoon and Malecki, 2009).

Our study extends work on user entrepreneurship by defining core constructs and explaining the commercialization patterns of users who create or enter new markets in different industries. Effectual strategies that assume that opportunities emerge when created by an entrepreneur have been positively associated with venture performance (Read et al., 2009). We pay special attention to the strategies users follow to remedy legal uncertainty when applying borrowed assets, the new knowledge they need to acquire, and the support they receive from their community of peers. Based on case studies, we inductively generate a model describing key elements of the strategies user entrepreneurs follow when commercializing products or services in new and economically relevant markets.

After introducing our research design, we describe the relevant cases and present the results of our study in the form of descriptive propositions. We conclude with a discussion of our findings, the implications for research, management, and policy, and outline a future research agenda for user entrepreneurship and strategy.

SAMPLE AND RESEARCH DESIGN

Machinima offered an ideal context to explore the research question of this paper because (1) user entrepreneurs could be observed entering the animation industry[7] over the past ten years

while the genre was in the process of emerging; (2) the animation industry was traditionally characterized by high entry barriers, leading to "creative" entry strategies; (3) Machinima production involves proprietary software as well as artwork, allowing a nuanced observation of how users manage IP conflicts; (4) users frequently possess advanced gaming skills but need to acquire film production knowledge to sustain a business in the target industry; and (5) users display high variance in their entrepreneurial approaches over time, ranging from product sales to diversification into consulting, software development, and online services.[8]

Our research comprised three phases: case sampling, data gathering, and data analysis. We conducted a multiple, non-embedded case study (Yin, 2003), and gathered data from seven firms within the Machinima community representing the entire population of Machinima-based businesses at the time of this study. We follow an inductive logic to theorize about industry entry and market fragmentation by user entrepreneurs and generate propositions derived from the cases (Cohen, 1980; Eisenhardt, 1989; Eisenhardt and Graebner, 2007; Glaser and Strauss, 1967; Strauss and Corbin, 1990). Theory development on a small sample size has been discussed by March et al. (1991), Eisenhardt (1989), and Siggelkow (2007), while exemplary works with the methodology include Vaughn (1990), Lawrence et al. (2002), and Pervez et al. (2008). Eisenhardt (1989: 545) proposes "a number between four and ten cases to usually work well" and allows for sufficient complexity without creating too much data. Such a sample size allows researchers to describe and analyze the cases in a systematic and methodical manner, leading to thorough contextual interpretation.

2.1. Sample

The production of animated films used to be restricted to media professionals who could afford the expensive software packages needed. These restrictions led users in the animation industry to produce films with games, inspired by innovative gamers who developed methods to record their game play. Such games are relatively cheap compared to traditional production tools. In addition, most of the in-game assets, like characters and landscapes that resemble actors and scenes, are already available, thus reducing overall production cost and time.[9]

Defined as "shooting film in a real-time 3D environment" (AMAS, see footnote 6), Machinima is (1) a production technology, and

(2) the name for the genre. It is deeply rooted in the gaming culture where gamers, early on, experienced the need to record, edit, and distribute proof of their gaming skills on film to demonstrate their proficiency as gamers. Adding story elements to their films (FK—see Appendix for full names and affiliation of interview partners), Machinima users in the animation industry later introduced a new genre that can be clearly distinguished from traditional animation (Mezias and Mezias, 2000; Peretti and Negro, 2007) when they produced low-cost films for themselves or close friends (Morris et al., 2005). The unique characteristics of the Machinima production process enabled users to become entrepreneurs, applying gaming technology in the animation industry as well as related industries, like film distribution and production support. The cases in our sample cover all aspects of the product development and commercialization process.

[7] According to the International Standard Industrial Classification of All Economic Activities, Rev.4 (ISIC), software game development (6201) belongs to another industry than motion picture production and distribution (5911, 5912, 5913, and 6020, respectively). The first two digits are sometimes used to denote a specific industry. (See Farjoun, 1994, for a discussion of the relatedness of industries in terms of knowledge, an aspect we use to define complementary assets and show the market entry.)

[8] A number of publications (Marino, 2004; Morris et al., 2005; Hancock & Ingram, 2007) cover the topic of Machinima for the general reader.

[9] Estimates of costs alone show that Machinima production amounts to a fraction of animation production. See also an article by the BBC http://news.bbc.co.uk/1/hi/technology/7045018.stm (October 26, 2007).

Company	Strange Company	ILL Clan / ESC	Machinima.com	Fountainhead Entertainment	Rooster Teeth Productions	Short Fuze	Bong + Dern
founding year	1997	1997 / 2007	2000	2000	2003	2003	2005
Founder(s)	Hugh Hancock Gordon McDonald	Paul Marino Frank Dellario Matt Dominianni Manu Smith	Hugh Hancock	Anna Kang	Burnie Burns Matt Hullum Geoff Ramsey Jason Saldaña Gustavo Sorola	Dave Lloyd Matt kelland	Chris Burke
Value driver	Consulting/Book	Second Life Media	Web Platform	Music Video/Machinima Tool	DVD/Sponsoring / Merchandising	Machinima Production Tool	Commercial videos
Machinima movies / series	*Ozymandias* *Eschaton* *BloodSpell*	*Life Performances* *Larry and Lenny* *Tra5hTa1k*	*Pathfinders*	*Zero7 - In the Waiting Line* *Anna*	*Red. vs. Blue* *The Strangerhood* *PANICS* *1-800-Magic*	*James Bond: No License*	*This Spartan Life*
Currently producing movies	Yes (to demonstrate capabilities)	Yes	Yes (offering content on their website)	No	Yes	No	Yes
Other non-movie contribution	LithTech Film Producer (Machinima tool)		Forum Webhosting	Machinimation		Moviestorm	
Business with Machinima	Non-movie	Movie	Non-movie	Exit	Movie	Non-movie	Movie
Location	Edinburgh UK	Brooklyn, NY USA	Los Angeles, CA USA	Mesquite, TX USA	Austin, TX USA	Cambridge UK	New York, NY USA
Financing	Bootstrap	Transition to ESC	Advertisement	Self-financed	Self-financed	2.8m USD venture capital	Venture capital/self-financed
Employees	> 2	5 / > 50	8	Undisclosed	6	> 3	2
User Experience	*Hugh Hancock was a Quake Gamer*	*ILL Clan was originally a Quake Gaming Clan*	(see Strange Company)	*Game background: affiliation with id*	*Avid gamers - did game reviews before*	*Produced mobile phone games*	*Halo Gamer*
First release (phase one)	*Eschaton* , 1999	*Apartment Huntin'* , 1997	Platform launched in 2000	*In the Waiting Line* , 2003	*Red vs. Blue* Episodes 1-12, 2003	*James Bond: No License* , 2004	*This Spartan Life* , 2005
Initial commercialization (phase two)	Commissioned machinima based on critical acclaim	Commissioned machinima based on critical acclaim	Advertisement / financed through Strange Company	*Game over* for United Paramount Network	Merchandise for *Red vs. Blue* , 2003	Book publication: Morris, Kelland, and Lloyd, 2005	Commissioned mchinima based on collaboration with the ILL Clan
Revenue model as of 2007	Consulting, book sale	Footage for Second Life (workforce hired by ESC)	Advertisement	Machinima business discontinued	DVD, merchandise, subscription fees	Moviestorm add-on sales through packs	Unclear

Table 2: Sample of firms for the cases

Every important step in the history of Machinima was initiated and conducted by users who played games and experimented with the Machinima production technology. Some of these users went on to form firms, including Rooster Teeth Productions, the ILL Clan, Strange Company, Machinima.com, Bong + Dern, and Fountainhead Entertainment. Table 1 provides an overview of the samples in this study.

Interviewees brought these seven firms, which we later identified as the population, to our attention as predominant examples of Machinima-based commercialization. Thus, we defined "successful" Machinima firms according to informants (Brown and Eisenhardt, 1997). All seven firms had produced Machinima films and won at least one award at an AMAS Film Festival.[10] Three started by producing animation and continued to do so; another three altered their sources of revenue after having produced Machinima; and one firm, Machinima.com, started producing animation later on to supplement their portfolio. Thus, all firms once entered the animation industry. Of the three firms that altered their sources or revenue, one chose to turn away from Machinima. We conducted literal and theoretical replication (Yin, 2003) by considering firms with operations in the animation industry as well as firms that developed alternative sources of revenue different than, but still based on, the Machinima production experience. The fact that user entrepreneurs show a tendency to display their capabilities at championships (Baldwin et al., 2006), like the annual AMAS Film Festival, provides further confidence that our sample covers the entire population at the time of data gathering.

Within the community, the visibility of Strange Company, the ILL Clan, and Machinima.com was extraordinarily high due to their involvement in the formation of Machinima. Rooster Teeth Productions was the prime example of a firm that generates revenues solely from the production of films and related products common to this industry. Fountainhead Entertainment used to be a Machinima pioneer in producing animation as

[10] A film festival held by the Academy of Machinima Arts and Sciences (AMAS)

well as supplying a production tool.[11] It originally focused on a Machinima-centered business and then refocused its activities on mobile phone games. Short Fuze first released the film Desert Combat: James Bond—No License in 2004; later, Matt Kelland and Dave Lloyd wrote the first book describing Machinima for the general reader (Morris et al., 2005). As a result of technical problems experienced during the production of their film and their observation of the community's needs, Short Fuze began to develop Moviestorm, an easy-to-use Machinima production tool, for which they received £450,000 in seed funding in 2005, followed by a first round of funding of £950,000 in 2007. This Spartan Life, produced by Bong + Dern productions, is an award-winning virtual talk show that received substantial publicity because of its high-profile guests from game companies.

Although the Electric Sheep Company (ESC) hired the workforce of the ILL Clan in 2007, we regard the ILL Clan as a separate entity in our sample for three reasons: (1) the ILL Clan had been producing Machinima since 1997, that is, for most of our observation period;

(2) the founders retained their brand and continued to make their existing and new animation products available under this label; and (3) the group continued to produce Machinima for ESC, which operated in virtual worlds such as Second Life.

The resulting sample combines all incorporated Machinima businesses that were at least a year old. Their common denominator is the entry of its founders into the animation industry as Machinima users at one point in time, ultimately leading to commercialization. The firms differ in terms of their commercialization activities, financing, size, age, ambitions, and goals. We also included one firm that discontinued its Machinima-related revenue source.

2.2. Data gathering

Data gathering took place in five phases, including both real-time observations and retrospective data (Brown and Eisenhardt, 1997). Desk research delivered insights on how the user community defined itself, who participated, the motives of the different users, and (most importantly) active firms as well as those that had ceased operation. Understanding the phenomenon helped us establish more effective relations with key informants. Hosting Machinima films, we retrieved plenty of information from community websites[12] including the names of the producer, the director, and the year of publication. Moreover, some of the films and credit files provided information about the production process, individuals, and firms involved. Altogether, we read 32 articles, studied a 340-page report on the video game industry, watched more than 100 short films, and browsed roughly 50 web pages to gain a thorough understanding of the phenomenon (for the use of rich information sources, see Vaughan, 1990).

Second, one of the authors participated in a four-day Machinima workshop to conduct field observations and build relations with the Machinima community. The workshop covered the entire Machinima production process and the author created a film to gain first-hand experience of the process.

Third, in November 2006, we identified interviewees from a variety of backgrounds and began semi-structured interviews. This approach allowed us to react to replies and adapt the questions to a candidate's profile. We usually started the interview with questions about an individual's background and education to decide later on whether the interviewee was an objective observer or a key informant on a topic. The initial set of questions, based mainly on desk research, was tested during the workshop. The questionnaire was subsequently refined and tailored to the specific background of each interviewee, based on information taken from online résumés or previous interviews.

Fourth, before entering the second round of interviews we analyzed our preliminary results. Having a general understanding of Machinima and the community that supported it, we focused on user entrepreneurs, some of their legal advisors, user innovators who played an important role during the development of Machinima—people who provided valuable background information, and finally games companies to complement the picture. From February to March 2007 we completed seven interviews, one face-to-face, and another six conducted by telephone (see

[11] Machinimation is a real-time 3D filmmaking software add-on representing a dedicated solely to Machinima. modification (mod) of id Software's Quake III Arena.

[12] Such as www.Machinima.com, www.mprem.com (Machinima Premiere) or www.gamevideos.com.

Table A2 in the Appendix for a full list and description of interview partners).

Fifth, another author travelled to New York and Texas and conducted a total of ten interviews and participant observation during a week on site with Rooster Teeth Productions, the ILL Clan, Fountainhead, and an ESC representative. Daily records of working routines, the Machinima production process, and other office tasks were kept; notes were taken while attending meetings, lunch, and evening events (Brown and Eisenhardt, 1997). Five formal interviews were conducted, three of which were recorded on video for later analysis and classroom use. In addition, more than 100 photographs of Machinima working environments were taken during the field trip.

In total we conducted 25 interviews, 21 in English and four in German, each lasting 55 min on average. We transcribed 19 interviews (20–30 pages each) verbatim. In addition, we cross-checked relevant information with other interview data or facts from desk research incorporating external links and comments in the text.

2.3. Data analysis

We compiled individual case studies, based on the data gathered from the five phases. First, bearing in mind the process of traditional user entrepreneurship from existing literature, the data were prescreened to derive a common coding scheme (see Appendix). Our interviewees' frequent and unexpected references to IP issues and legal uncertainty indicated that the entire process of commercialization was greatly influenced by the legal aspects of game engine use. Hence, we considered relevant literature in this field to support the coding scheme.

A rough sketch of what industry entry involved supported us in describing the phenomenon to interviewees. We then coded the transcribed interviews using MAXQDA, a software tool for text analysis. While analyzing the interviews, we coded statements in the text, which allowed us to sort and evaluate information. Two researchers working in parallel conducted the coding. After the initial coding, results were merged and the second coder recoded selected interviews, con-

Firm	Strange Company	ILL Clan / ESC	Machinima.com	Fountainhead Entertainment	Rooster Teeth Productions	Short Fuze	This Spartan Life
Proposition 1 *Impact of game companies' preference towards sharing on access to complementary assets*	Use of game engines (e.g. Neverwinter Nights)	Scripting work in *Quake* (LMPC), later engine acquired and creation of own artwork: Low dependence on freely available resources	No direct production of machinima	Affiliation with id Software (producer of *Quake*) and access to development resources	Agreement with Bungie (producer of *Halo* and Microsoft subsidiary until October 2007)	Machinima produced with *Battlefield 1942*, later independent of game companies: Moviestorm consists of entirely new artwork	Agreement with Bungie
Support for P1	+	+	Not applicable	+	+	+	+
Proposition 2 *Impact of complementary assets on commercialization*	Use of existing game engines but creation of own artwork	Use of existing game engines but creation of own artwork	Not applicable	Complementary assets did not lead to commercialization	Heavy use of existing artwork and game engines for most if not all Machinima	Reliance on in-house complementrary assets (Moviestorm)	Heavy use of existing artwork and game engines
Support for P2	+	+	Not applicable	-	+	+	+
Proposition 3a *Impact of the communitiy on the skills to apply complementary assets*	Pioneering role within the community of machinimators: co-founder of the AMAS	Pioneering role within the community of machinimators: co-founder of the AMAS	Central platform for the machinimator community (distribution and exchange): co-founder of the AMAS	Pioneering role within the community of machinimators: co-founder of the AMAS	Garage approach: outreach only later	Weak link. Investors provide knowledge and industry access	Inspiration and learning from fellow machinimators
Support for P3a	+	+	+	+	+	-	+
Proposition 3b *Talent pool of the target industry as preferred choice for domain knowledge acquisition*	Founders already had the knowledge from the target industry	Founders already had the knowledge from the target industry	Platfrom attracted members of the target industry to contribute	External director only affiliated for time of project	New team members were hired who worked in Hollywood	Careful balance of management team to include relevant skills	Prior production knowledge from previous engagements
Support for P3b	-	-	+	+	+	+	+
Proposition 4 *Impact of knowledge combination on commercialization*	Education in film and experience in film and audio arts to complement user innovation	Extensive experience in animation to complement user innovation	Growth of the community (machinimators and audience) enabled Machinima.com to	Alliances with film directors, later exit	Motion picture industry experience in the team (e.g. Matt Hullum) combined with user innovation	Moviestorm was designed to enable the simple creation of Machinima by everybody	Education in film and experience in audio arts
Support for P4	+	+	+	+	+	+	+

Table 3: Overview of propositions and grounding in cases

tributing to inter-coder reliability.

Applying an iterative process with an overlap of data analysis and data collection (Eisenhardt, 1989; Glaser and Strauss, 1967), we used the distilled interview data, including the codes and the higher code categories (e.g.: 2 Domain knowledge, 2.1 Education, 2.2 Work experience, 2.3 Prior film production), as well as secondary information, to refine our case studies by gathering additional data whenever gaps were identified during coding. We evaluated the interview data and compared coding of key informants (e.g., director) with objective observers' (e.g., lawyer) views where applicable. We triangulated the findings from the interviews with our observation records (Pettigrew, 1988), independent information gathered from the Internet, image, and video material, and third-party newspaper articles. We produced detailed case write-ups for each firm to cope with the magnitude of data. While structuring and analyzing the within-case data, we checked whether information depicting every construct was obtained for each case. This led us to drop one early construct but to formulate another, enabling further consolidation of the coding scheme. The final case study write-ups facilitated the comparison of all ventures' positions in terms of user entrepreneurship. Searching for cross-case patterns enabled us to extract the general sequence of actions users followed to commercialize their products or services as well as to identify the different stakeholders or stakeholder groups involved in the different phases of the commercialization process (Eisenhardt, 1989). Based on the flow of actions, we derived five propositions and drew up a table summarizing the findings for each (see Table 2). To enhance construct validity, all propositions were compared and discussed in the light of the existing literature. The maintenance of individual case studies during the analysis supported replication across firms. A final working paper was sent to all informants for comments and feedback, which were integrated with the text.

TOWARD A PROCESS MODEL OF INDUSTRY ENTRY BY USER ENTREPRENEURS

In this section we present findings on how, in the case of Machinima, users become entrepreneurs who commercialize their products or services based on assets from another industry in a two-phase process. We inductively develop a model from the cases and formulate five propositions (Table 2) to explain user entrepreneurs' behavior and the consequences of their activities (Fig. 2, explained throughout the text). We emphasize the interactions between user entrepreneurs and firms that selectively share complementary assets, between user entrepreneurs and their community of peers who exchange knowledge about the application of complementary assets, and between user entrepreneurs and talent from the motion picture industry to acquire domain knowledge. First, we describe the context of industry entry, which occurred in two phases.

3.1. Under the radar of incumbent firms: a two-phase process of industry entry

The first phase of industry entry consists of a lateral move from one industry to another under the radar of incumbents: When first applying video games as tools to produce animated shorts, users emerged from the domain of the video game industry to enter and become producers in the animation industry where they used games as alien production tools—most of them continuing to be avid gamers. Neither companies in the video game nor in the animation industry paid attention to or followed the activities of these "hobbyists" who operated on a very small scale of product development (cp. Depoorter, 2009), which we consider under the radar. At the same time, video game companies were harsh with users who altered or modified video games regardless of the scope of modifications or their commercial intention (AC, CB). The same holds true for film production companies that saw their media assets re-used and or distributed over the Internet.

The following quote from Geoff Ramsey (GR) illustrates how the founders of Rooster Teeth started out to reach an audience from a spare-time activity and moved on to a full-time activity, following the success they achieved in terms of audience and attention received.

"Initially we worked in Burnie's spare room in Burnie's house. And it was a small room [...] and it wasn't so bad when we were just Burnie and I. But you know, two people turning into five people in a room that's maybe [...] 250 square feet became pretty cramped, and at some point Red vs. Blue [...] became successful enough, where I quit the day job and devoted full time to Red vs. Blue. Burnie eventually did it as well." (GR)

While one might argue that users are not located in a specific industry without commercialization, they nonetheless produce and distribute animation. This compares to cases of open source software, where no commercial intentions may exist but where market share of incumbent software companies is affected. Geoff Ramsey's anecdote provides a typical illustration of the transition from phase one to phase two.

"Machinima was a very cheap way to produce animation or, you know, any kind of narrative, but as it becomes more popular the cost increases. And, unfortunately, the more popular you are the more money you generate ...[and the more] bandwidth you're paying for people to be able to watch your series. ...So we were always looking for ways to help offset costs. So, I reckon June of 2003, this is around episode 12 of Red vs. Blue, I put a thread in the forums: I said, if I made a Red vs. Blue T-shirt, just a white T-shirt with a logo on it, would you be interested in buying it? And the response was overwhelming. And so I made a T-shirt and people bought it." (GR)

In phase two, user entrepreneurs experimented with various ways to commercialize their ideas. Rooster Teeth introduced merchandising to help offset costs, besides DVD sales, an increasingly popular revenue window in the motion picture industry in times when box office sales do not suffice to break even provided the production costs (Eliashberg et al., 2006; Wasko et al., 1993; Yoon and Malecki, 2009). Table 1 summarizes the activities undertaken by the firms in our sample during under-the-radar entry to the animation industry (phase one) and first opportunity recognition and commercialization strategies (phase two).

It is important to note that the user entrepreneurs were the producers as well as the distributors, exhibitors, and the licensees of their own merchandise—roles usually separated along the value chain or at least split into various subdivisions of

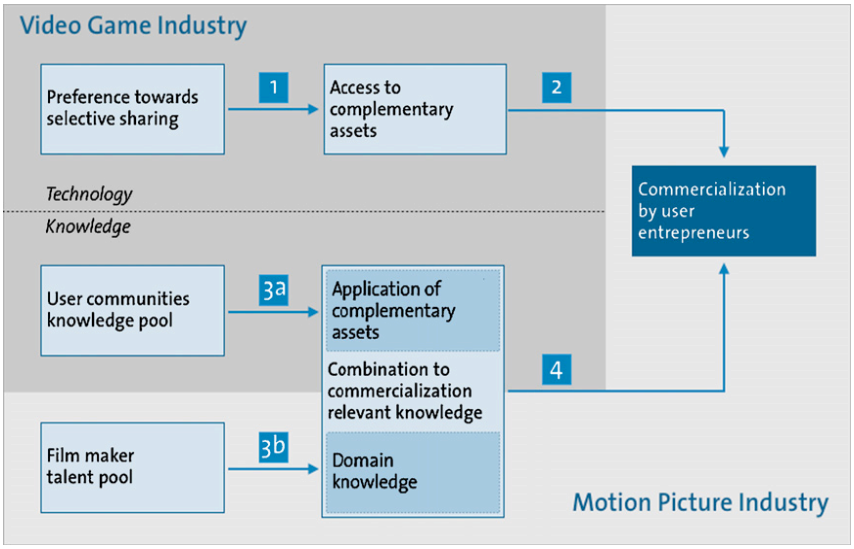


Figure 2: Process model of commercialization by user entrepreneurs across industries

corporate groups in the motion picture industry. Thus, user entrepreneurs in our case entered several related industries subsequently based on potential business opportunities they recognized during display of their core product, the Machinima films.

Wendy Selzer (WS), of Brooklyn Law School, explained the critical legal situation in which user entrepreneurs found themselves when entering phase two. Here, their foremost concern is the uncertainty regarding a potential legal dispute with the game companies:

"They typically ask that question when they're trying to do ... when they move from the non-commercial into the commercial space because another of the factors in the fair use inquiry is whether your use is commercial or non-commercial as well as that somebody is far less likely to sue you if you're just doing non-commercial. Or at once you start making money that they see it as something that they might be getting a cut of. So at that point, you see more of them [user entrepreneurs] asking the question." (WS)

Operating within uncertain legal boundaries, resulting from the copyright restrictions that came with the video games, made access to distribution channels anything but easy. TV networks exerted pressure on user entrepreneurs by passing on the legal clarification regarding copyright infringement (WS). User entrepreneurs in our sample (AK, MK, CB, PM) reported having experienced conservative reactions to Machinima from industry incumbents such as TV networks and film studios. This negative attitude meant they had either to seek alternative sources of revenue or obtain explicit permission from IP holders.

3.2. Asset holders' preference for selective sharing of proprietary assets (industry A)

Dependency on video games as complementary assets for animation production (see Teece, 1986) complicated industry entry by user entrepreneurs for a number of reasons. Game engines (movements in 3D space, sound, artificial intelligence, look-and-feel, etc.) and artwork (settings, scenery, characters, skins, textures) represent significant investments by companies in the game industry. Copyright law and EULAs protect these assets, prohibiting any (commercial) application of the acquired video game or the artwork that comes with it. These copyright agreements made it difficult if not impossible for user entrepreneurs to sell their Machinima films directly (see also Marino, 2004; Hancock and Ingram, 2007). Fred von Lohmann (FvL), IP lawyer at the Electronic Frontier Foundation, clarified the difference in copyright for the two main components of a game, the engine and the assets. While copyright on the assets is closely defined and attributed to the creator, the output generated by the game engine is far from easy to classify and thus vulnerable to lawsuits, giving user entrepreneurs little leeway to negotiate with distributors:

"I'm sure a game company would say, 'We own copyrights in all of the graphics that comprise the game, so the character designs, the textures, the landscape arrangements,' you know, all of the graphical elements. I'm sure they would claim that there's a copyright there, and they may also argue that they have a copyright in the engine and the output of the engine is, you know, therefore a derivative work. I think that is a more far-fetched argument, that argument would quickly lead to the conclusion that Microsoft owns everything that is produced with Microsoft Word, and I think that argument is far fetched but, again, it's not inconceivable. So those are the two elements that are most likely to arise in a copyright dispute: the graphical objects themselves and then the actual ...the algorithm—the engine that puts it all together." (FvL)

Using games for animation production thus requires user entrepreneurs to gain legal access to the pertaining assets. Chris Burke of Bong + Dern explained his approach to third-party IP as reaching an agreement with the game company to avoid any risk, despite incurring high transaction costs originating from the negotiations:

"Generally the intellectual property rights are always a big restriction, you know, even if the game company is willing to work with you, it's still going to take you six months, sometimes a year to work out something which will allow you to do, what you want to do." (CB)

Users solved the dilemma of uncertainty in two ways: they either negotiated a contract with the game company who granted use and thus selectively shared game assets for clearly defined purposes, or they remodeled large parts of the game. While the ILL Clan and Strange Company applied specialized tools to "re-skin" avatars and avoid using existing artwork, Rooster Teeth struck a deal, which allowed them to use the artwork from Halo for their Red vs. Blue franchise. We discuss the advantages and disadvantages of either strategy later. Burnie Burns of Rooster Teeth described their relationship with Microsoft as being grounded in excitement about the possibilities user entrepreneurs discovered, while at the same time showing generosity and interest in a long-term engagement:

"[Microsoft] what we found was we found a group of people who really like innovative cool stuff and they saw something that I think they thought was unique and they worked with us and it was really great, it was surprising. I was really amazed and I continue to be amazed at how great they are to us and how much freedom they give us and how long we are able to work with such a big company." (BB)

Selective sharing by game companies takes different forms: they license assets under creative commons or open source (OS) licenses (such as parts of the Unreal Tournament engine released by Epic Games in 1999[13] or the tool set for Neverwinter Nights by Bioware); or they close contracts with Machinima producers that allow user entrepreneurs to use the games (engines and/or artworks) for commercial purposes. The game companies' incentives to grant these exceptions to the standard licensing terms include the promotion or advertisement of games through Machinima (AC, BB, FvL, PD). Phil DeBevoise (PD) of Machinima.com reported:

"The game publishers recognize this is being a terrific free promotional and marketing vehicle for their games and they are very much, you know, we have spoken to pretty much all of them ...are very excited, and very much want to engage the gaming community and have them be, you know, very much invested in their games. Like when they are making Machinima, that's a true act of love and passion. They are working many hours not only playing their game, but they're also making this." (PD)

Without the game companies' consent, users face the dilemma of producing films without permission to distribute and run legal risks of infringing the copyright or the EULA. To date, no case involving Machinima had been tried in court nor have EULAs with special Machinima clauses spread widely. One game publisher that recently engaged in encouraging its users to produce Machinima under terms of fair use is Blizzard Entertainment, the producer of World of Warcraft.[14] In general, the right to produce Machinima under fair use of video games (U.S. copyright law), as users hope, remains uncertain and a sense of frustration about this dilemma is widespread in the Machinima community (voiced by all interview partners).

Another approach was to create large amounts of artwork from scratch. The ILL Clan pioneered this work with their film Hardly Workin' for which "we changed the entire look of the game" (PM), accomplished by two members during two years of part-time work. Shooting the film, which involved five people, and integrating the improvisors' dialogues, took a further six months. Creating own assets by modifying the game gave confidence of not infringing any IP rights, as Paul Marino (PM) and Frank Dellario (FD) described:

"Quake allowed you to re-skin your characters, that was one of the things that id Software did ...that was a very innovative approach to customizing the game and [...] we used this feature to do that for our film." (PM)

"Using Quake 2 we then made Hardly Workin' which Paul Marino directed and that we created all on our own assets. The only thing we used was the engine. The map, everything we used, we created from scratch, cause we said 'Let's get away from other people's IP.'" (FD)

Four firms in the sample (The ILL Clan, Strange, Fountainhead, Short Fuze) opted for the costly

[13] Ports to Linux are available from the Open UT project: http://openut.sourceforge.net/info.php.

[14] http://www.wow-europe.com/de/community/Machinima/letter.html.

approach of creating artwork themselves. Bong + Dern and Rooster Teeth were the exception, in that they primarily used game assets after negotiating with the game company. Since the contract details between game companies and Machinima producers remain undisclosed, this study cannot conclude whether this involved great cost or whether the companies placed their assets at the disposal of selected user entrepreneurs. However, the evidence shows that support tends to flow from game companies to users, in terms of permission to publish, submit to film festivals, contract work (commissioned Machinima, TV advertisements and game commercials), and feature support. The Sims 2 features integrated video capturing and Blizzard recently added tutorials for Machinima in World of Warcraft. The latest sequel of Halo offers a save film feature, which allows players to view and record their game play in retrospect from camera angles they did not use during actual game play. This also demonstrates that the Machinima community, after gaining enough leverage, had a reverse impact on the game companies (compare von Hippel, 1988, on producers who incorporate user innovation).

Thus, we propose (1): *If firms (in other industries) show a preference toward selective sharing of proprietary assets, this positively impacts the user entrepreneurs' access to complementary assets for commercialization.*

User entrepreneurs often first use complementary assets regardless of potential IP infringements (phase one) and later recognize an opportunity to commercialize (phase two)—a central finding in the user entrepreneurship process. The failure of asset holders to approve the commercial application of complementary assets may represent a road-block for user entrepreneurs, since access to complementary assets is critical for commercialization, even if access is granted ex post (the initial use).

3.3. User entrepreneurs' access to complementary assets for commercialization (industry B)

User entrepreneurs need access to a range of complementary assets, including those of game companies, as we discussed in the previous section. We now relate these assets to commercialization. The process of finding and locating relevant assets in the video game industry to match with a story and produce a film is illustrated by Burnie Burns (BB) of Rooster Teeth:

> *"I think, probably since Red vs. Blue started, we've never had an idea and then found a game to make the idea work. That's how Red vs. Blue started, we had the idea for Red vs. Blue and we found Halo as the way to do it. Everything else has been where we've had a game with great tools and great Machinima capabilities and then we have found a story to match the theme of the game. We try not to make it about the game but thematically it makes sense, if you are going to have elements in the game already, we want to include them in the story. You can't take a game like F.E.A.R. and turn it into a romantic comedy, well you could but you know...It's more of a suspension of disbelief in that case. It's more about just position this. We try to do stories that are believable, within the context." (BB)*

All firms in the sample used and relied on complementary assets such as game engines, tool sets, and artwork stemming from video games, which were incorporated in the production process. In some cases, the artwork even stimulated the innovation process (BB).

Following a different approach, Short Fuze relied on complementary assets they produced themselves: a software product called Moviestorm. The firm's business does not rely directly on the application of game assets but on the user experience associated with them. After seizing the opportunity while active in the animation industry, Short Fuze created a game-like environment for users to shoot films, thus actually supplying complementary assets to users. Matt Kelland (MK) of Short Fuze depicts legal uncertainty as the incentive to create Moviestorm, which substitutes the formerly used video games:

> *"One of the biggest issues with Machinima is that there is a big debate about copyright issues, because when people are making videos, using games made by professional game companies, they say, well, you know, you're using our sounds, you're using our assets, you're using our animations, our levels and so on. And everybody in the Machinima community is waiting for the first big lawsuit to happen and one of the things we decided was just to move around this by saying, well, we intend to own the engine, the assets, maps and everything so when somebody makes a movie with Moviestorm, they own it. In just the same way that they would own anything they created with Word or Photoshop." (MK)*

Fountainhead quickly ceased to commercialize animation despite its access to complementary assets that allowed it to create groundbreaking Machinima early on. Machinima.com stands alone, due to its special role as a web platform

supporting the community[15] in various ways, ranging from hosting to education. The remaining four firms, however, lend support to our second proposition by relying on complementary assets for their commercialization strategy.

Thus, we propose (2): *User entrepreneurs' access to complementary assets positively impacts commercialization.*

Proposition two also reflects the two-phase process, since access to complementary assets is provided by user innovators, whom we regard as separate from user entrepreneurs, as well as the game companies' willingness to share these assets.

For completeness, it should be noted that while complementary assets play a crucial role in creating Machinima, they do not

replace the creation of original artwork like voice-over and audio effects by user entrepreneurs. The integration of assets, artwork, and potential post-production with the actual shooting requires considerable skill and experience in making Machinima all in areas of domain knowledge, which we turn to next.

3.4. Acquisition of new domain knowledge

Critical new domain knowledge for commercialization represents the qualification necessary not only to produce but also to distribute animation shot in video games that appeals to a greater audience. It includes education and experience in cinematography, the creation of a narrative, screenwriting, post-production and editing skills, as well as knowledge of the film business such as marketing, reaching an audience, creating sequels, and sustaining the interest of an audience in film characters and stories—in other words, the creative skills that are crucial to producing any kind of film, including Machinima. The application of complementary assets refers to the users' technical skills at applying and exploiting given features in a video game, such as specialized tools to reshape the appearance of characters, puppeteering, or the capturing of raw video material, basically all functions related to execution.

Friedrich Kirschner (FK), member of the AMAS, explained early approaches to production knowledge exchange as being part of the maturing Machinima community. He describes the interweaving of technology and knowledge about the motion picture industry:

"Presenting [Machinima] to the community is less important. [...] early on it was about [technology]. That has changed a little today because the community moved to a field that is less concerned with the technology. A few years ago, technology was more central ... to consider things from a technical perspective. There were different games one could have possibly worked with. Then pros and cons were discussed, that was definitively important, because back then in the Machinima.com community one didn't come from the game but from the wish to produce a movie. And because of that, one could be talked into using another game or at least checking out another game, because it was used as a tool. [...] In this regard, Machinima matured a little or at least the community thinks it matured a bit. And this has more to do with how they deal with the newcomers. [...] Machinima producers simply don't know anything about the 180-degree rule and cutting and directing in the broadest sense. There, they try to very much catch up and tutorials are written and hints are exchanged and that kind of stuff." (FK, translated from German by the authors)

Knowing each other's work well, users exchanged experiences and opinions during film festivals (AMAS, Sundance, Tribeca, Bit-film, and others), game conferences, and via online discussion forums (Machinima.com and others). Some even wrote Machinima beginners' books about their specific production knowledge (Hancock and Ingram, 2007; Marino, 2004) or kept regularly updated blogs (FK, PM, FD). Key contributions by community members in the form of tools (Uwe Girlich's Little Movie Processing Center or Friedrich Kirschner's Movie Sandbox[16]) altered the way users worked and frequently facilitated the art of Machinima for new talent. Certain Machinima pieces reverberated inside the community and inspired new work because of their demonstrable feasibility. The way tools were applied was critical for the development process since it would shape the final product and, ultimately, the extent of commercialization. A global community of users sustained discussions about the most effective ways to use tools, both legally and artistically. Chris Burke (CB) of Bong + Dern exemplifies this stimulation of the community with extraordinary contributions:

"Something that really fascinates me is non-narrative Machinima, which is, I guess, what grew out of what

[15] Because www.Machinima.com did not start commissioning films until the end of August 2007, after we interviewed their CEO Philip DeBevoise, we have not considered Machinima.com for some of our propositions.

[16] http://www.moviesandbox.com/

used to be speed run and all that stuff [recording of game play]. Things like Warthog Jump. [...] I think that's fascinating. To me that's something you would never have in any other medium. And some users they watch it and they think, oh it's just some guy messing around, which it is. But there's, I think there's a really ... there's like real theory going on behind that." (CB)

These observations demonstrate that another indispensable aspect of the user entrepreneurship process in this case involved the community of users (Machinimators), which impacted positively not only on domain knowledge from the motion picture industry, but also on the combination of domain knowledge with complementary assets. While the community of Machinimators helped to promote the art of producing high-quality animation and provided the resources to develop and deepen domain knowledge, it also promoted the integration and application of complementary assets.

Burnie Burns (BB) and Geoff Ramsey (GR) of Rooster Teeth produced the first series of Red vs. Blue in their homes while voice actors called in over the phone to contribute their part of the scripts. Later, they recruited friends, some of whom had worked in Hollywood studios, to join the team. Domain knowledge proved crucial when it came to commercialization. Not only did Matt Hullum bring valuable film experience to the team, but the development of the series' characters and epic story elements over 100 episodes, commitment to the web as the preferred channel of distribution, as well as contracts that secured the rights to sell DVDs and merchandise, all represent thorough knowledge about the motion picture industry that few video game users possess.

The case of Fountainhead Entertainment demonstrates the contrary situation. Its founder, Anna Kang (AK), one of the co-founders of AMAS, created groundbreaking Machinima work. Despite critical acclaim, their efforts did not translate to sustained commercial activity in the animation industry, possibly because there was limited industry experience within Fountainhead—their critically acclaimed music video In the Waiting Line had been produced with an outside director who was interested in trying new technologies. The other Machinima producing firms in the sample had at least one core member or founder with an education in film or extensive industry experience in film or animation or both.

Despite the importance of domain knowledge, commercializing Machinima seemed to rely on the user entrepreneurs' ability to combine domain knowledge and experience with complementary assets. Users of video games approached animation production in new ways (Lowood, 2007; Marino, 2004) regarding their use of technical tools and cinematography, as Matt Kelland (MK) of Short Fuze commented. The crew at Short Fuze knew that there were important trade-offs when making animation with games or with user experience of gamers. Moviestorm, a game-like animation production environment, caters to users who want to produce Machinima but have little or no film-related education or experience:

"As a game player, my preconception is 'I don't have to do anything, the computer just works it all out for me.' And it may not be exactly what I want, but it's good enough and it was easy. Whereas an animator would say, well you have to be able to decide where they walk and how they walk and get it all absolutely right, which we say, it actually doesn't matter." (MK)

Our cases demonstrate that a community of users positively impacts on the accumulation of relevant domain knowledge from different industries. We can observe an important distinction between user innovators and user entrepreneurs. User innovators, within a community or by themselves, combine complementary assets (video games) with cinematography skill (e.g. Randall Glass with his film Warthog Jump, published in 2002, achieving effects once thought impossible). User entrepreneurs, however, take their animation products one step further and commercialize them in the animation industry. Depending on the user entrepreneurs' origin, the respective other domain knowledge has to be acquired.

To commercialize Machinima, user entrepreneurs need to combine domain knowledge with their skills at applying complementary assets. Consider again the example of Rooster Teeth: the tale of their successful series starts with the discovery of a bug in Microsoft's Halo game that allowed them to make their avatars look straight ahead while pointing their guns down. This non-feature enabled the dialogue scenes in Red vs. Blue. The deep experience with a video game, the cinematographic skill to exploit this bug artistically to create entertaining products, and the business knowledge of how to market the product need to come together for commercialization. All of our cases show that access to both kinds of

knowledge—domain knowledge as well as experience and skills—is necessary for commercialization.

We propose (3a): *The community of users positively impacts on user entrepreneurs' skills at applying complementary assets.*

The ability of all firms to generate revenue could be traced to talent from the motion picture industry, whether by temporary arrangements, hiring, or via the founders. We thus propose (3b): *Access to the talent pool of the target industry positively impacts on user entrepreneurs' acquisition of domain knowledge.*

Note that this finding is compelling in the sense that all but one user entrepreneur followed this course and phase one (under the radar) appeared as a prerequisite for phase two. Understanding how to produce Machinima film and the experience as users of Machinima tools, the insights gained from a community of peers, and the knowledge to be gained in the domain of the animation industry, all contribute to the possibility of entering phase two, commercialization.

In summary, commercialization was impossible, in our sample, without the combined knowledge of the domain of filmmaking and of the application of games as complementary assets to film production. We propose (4): *User entrepreneurs combine domain knowledge with the skills to apply complementary assets to commercialize their products.*

Table 2 gives a summary of the propositions and how they relate to the respective firms, with a short description of how each proposition is supported throughout the cases.

Discussion

Our findings relate to the emerging literature on user entrepreneurship which shares with an effectuation view on entrepreneurship theory the perspective that opportunities are created given a set of means (Read et al., 2009), here, the user experience. In this section we discuss where and how our propositions resonate with existing theory or depart from it.

Our finding that industry entry occurs in two phases extends existing work on user entrepreneurship (Baldwin et al., 2006; Shah and Tripsas, 2007) in that it separates the user innovator from the user entrepreneur while describing a coherent process of user entrepreneurship. A user innovator may enable phase one by extending the use of a product or technology (von Hippel, 2005; Faulkner and Runde, 2009), in our case enabling a video game to be used as a tool for animation production. Phase two may evolve without major user innovations if user entrepreneurs draw on an existing innovation for their own use and for later commercialization. Nevertheless, a deep familiarity with the use of the innovation was a prerequisite for commercialization for all entrepreneurs in our sample.

4.1. Propositions in the light of existing theory

Proposition 1. Some authors have argued that firms frequently solicit the use of their knowledge assets for a licensing fee (Chesbrough, 2003; Arora et al., 2001). Uncompensated tolerance of asset exploitation has been documented for cases of informal know-how trading (von Hippel, 1987), and in areas where participating firms do not compete directly and find it beneficial to support each other (Henkel, 2006; Dahlander, 2007). User entrepreneurs who deploy proprietary assets depend on the owner's tolerance of the application of these assets for commercialization. Our cases extend previous research on selective sharing and tolerance toward application by demonstrating the potential relevance of assets for user entrepreneurs' entry into commercial marketplaces. They also document a nascent group of firms that contribute to developing a new genre in the industry they enter, which underscores the novelty of their products and the associated risks of IP infringement.

Proposition 2. The innovation literature has so far generated only limited insights into the commercial use of selectively shared assets. We show that selective sharing may apply to different types of asset (game engine, artwork, toolkits, etc.), which in turn represent different trade-offs for the user entrepreneur in terms of gaining access to, or substituting for, the asset. We observe that while user innovators frequently make unauthorized use of copyright work (as described by Lee, 2008), user entrepreneurs display great sensitivity when working with others' copyright protected work. They either secure owners' explicit permission to create and distribute Machinima using video game artwork, or they completely re-create artwork to avoid conflict over intellectual property rights. Commercializing with others' assets in a new industry is a facet of the user entrepreneurship process not yet described by the literature.

Proposition 3. An important finding in the user innovation literature is that users tend to organize their innovation projects in communities (von Hippel and von Krogh, 2003; Shah, 2006; von Hippel, 2007). Members of these communities bring their individual domain knowledge to bear on technical problems, share solutions, promote their work, and develop and improve on technology (Franke and Shah, 2003; von Krogh et al., 2003; Jeppesen and Molin, 2003). As users who become entrepreneurs and enter a new industry, the firms in our sample provide a unique opportunity to observe where knowledge is sourced and to distinguish between knowledge of production methods, application or development of complementary assets, and domain knowledge. Little is known about this distinction in the user innovation literature, which is vague about when knowledge emerges from a user community and when it needs to be acquired through hiring, or as part of the founding team.

Within user communities, members share and exchange experience with peers from other knowledge domains (Lee and Cole, 2003; Spaeth et al., 2008). Kogut and Zander (1992) suggested studying how firms combine knowledge from internal as well as external sources for innovation (see also Schumpeter, 1934). Successful innovators need the ability to identify external knowledge as an important input to innovation and commercialization, and must have the capability

of combining new and existing knowledge (Alvarez and Barney, 2005; Brockman and Morgan, 2003; Chirico and Salvato, 2008). Users are considered a source of domain knowledge from ideation to complete product development (von Hippel, 1988, 2007; Baldwin et al., 2006; Ogawa and Piller, 2006; Füller et al., 2007); however, production methods are frequently considered proprietary (Henkel, 2006: 962). We identify the types and sources of knowledge combined by user entrepreneurs and propose ways in which a user community provides insights into the production technology for Machinima. By developing and evolving the use of video games for animation production, the community participates in entry into a new industry.

The firms in our sample exchange insights about video games in user communities (Cohendet and Simon, 2007) and hire market insiders to acquire domain knowledge. At the same time, they develop proprietary production knowledge, such as experience in narrative development and plotting, or post-production techniques. Revealing of knowledge by user entrepreneurs depends on the industry context, in that production tools—the application of game engines for Machinima production—are more liberally shared within the user community than production skills, such as cinematography and story development, key elements for revenue generation in the animation industry.

Proposition 4. When building new ventures in an industry, entrepreneurs need domain knowledge relevant to commercialization (Michael et al., 2002). When entering new industries, domain knowledge about target markets plays a key role in commercializing innovations. The case of Machinima shows that users' experience can translate into an ability to apply tools that serve as complementary assets in a new industry, and that a combination with domain knowledge enables them to commercialize their products. This finding departs from the literature on user innovation and user entrepreneurship, which assumes users' stronghold to be their domain of experience, their access to knowledge generated by their practice as users, as well as their network (Luthje et al., 2005; von Hippel, 2007). Our sample suggests that users' realm of entrepreneurial activity extends beyond the market where initial user innovation could be observed (the video game industry) to include other industries and the creation of new market fragments.

4.2. A summary of stakeholders and activities

In exploring the case of Machinima, we studied a new genre and a growing community of film and gaming enthusiasts. Table 3 provides an overview and summary of the stakeholders involved and their activities before and during the phases of commercialization described by our model. User innovation began before the users became entrepreneurs and the stories of their businesses unfolded. Our data cover the beginnings of user innovation in this field and the birth of the genre, since many of the individuals who went on to become entrepreneurs are today considered Machinima pioneers (Paul Marino, Hugh Hancock, Burnie Burns, Anna Kang, and others). They reported their first encounters with the technologies of recording and sharing game play, the creativity these spurred, and the first teams they formed to manage early productions.

In our account of the propositions, some stakeholders' positions were omitted for the sake of brevity. However, more detail about the roles played by the video game companies and the audience may facilitate the testing of our model in other contexts. We convey a comprehensive account of the Machinima case, reaching back as far as possible in its history, to open the possibility for others to discover and identify nascent markets and industries by recognizing user activity and behavior in terms of knowledge acquisition and re-interpretation of assets.

The columns in Table 3 show critical activities within the two relevant industries, video games and motion picture. The rows follow the stakeholders over time, going from top to bottom, from an enabling phase to user innovation and then to the two commercialization phases covered by our propositions. The relevant stakeholders, apart from user entrepreneurs, include gamers who are forming a community of Machinimators (the lead user community), incumbent firms in the video games industry, and consumers of videos. Shaded cells contain user entrepreneurs' key as described above, and may serve as reference points to compare with other stakeholders' activities.

	Industries / Stakeholders	Video Game Industry (A) Artwork & render engine of game	Motion Picture / Animation (Production) Industry (B) Machinima as production technology (Process) represent *Complementary assets*	 Motion picture production skills represent *Domain knowledge*	 Machinima films (Product) enable *Commercialization*
Enabling Phase	**Video game companies**	Offer customizable games under strict terms of use to satisfy customer needs in the video game market			
	Gamers (Lead users)	Experiment with new possibilities and develop skills in handling the game and designing new assets			
User Innovation Phase	**User innovators (A) / User (B)**	Develop new or modified games by creating new asset base (sometimes referred to as modding)	**Re-interpret the meaning of video games to become a complementary asset for animation production resulting in a new production technology: Machinima**	Mostly ignorant of cinematography	Produce first game-play recordings mostly to demonstrate the technique without commercial intent Innovate new genre: Machinima
	Video game companies	Stop user innovators from modifying and distributing modified *games* by legal means (cease and desist letters)	Generous, unaware, or careless about the application of their game engines as tools for animation production		
	Lead user community		Users establish Machinima.com as a community platform to disuss their experience. Free revealing of knowledge related to the production technology leads to an improvement of the overall process	Users discuss various topics centred around motion picture production in general to obtain basic domain knowledge about animation production based on Machinima	Publication and exchange of exemplary short films (e.g. Warthog Jump)
Commercialization Phase 1	**User entrepreneurs (*under the radar*)**		Develop advanced skills and improve process which they still freely reveal and openly communicate to fellow community members	Intensively acquire domain knowlege by either experimenting with the technology (learning through feedback from the community) or hiring people with domain knowledge	**Consistent publication of Machinima series to attract and retain audience; branding; tutorial publications by user entrepreneurs, first attempts at offsetting costs incurred through hardware and bandwidth; attracting venture capital.**
	Lead user community		Supports entrepreneurs with new production technology Establishes official entitiy: AMAS to promote and protect Machinima	Talent from the video production industry discovers the new genre and joins the community previously dominated gamers	Original customer base for entrepreneurial star-ups Provides feedback to new ventures
	Consumers (of Machinima films)			Critical consumption that can result in feedback offered via social software platforms	Provide feedback for further improvement of existing products (innovation relevant feedback like possible story-lline changes, etc.) Voice demand for new products and improved business models
Commercialization Phase 2	**User entrepreneurs *(becoming visible in the animation industry)***		**Managing legal uncertainty regarding the use of the complementary assets (in-game artwork and game engine use)** **Consult and employ lawyers to assist in negotiations with video**	**Commercialization depends on effective combination of domain knowledge by hiring talent from the motion picture industry**	**Far-ranging commercialization activities from the sale of Machinima to previledged online access as well as diversification into merchandising, consulting, hired work, etc.**
	Video game companies	Maintain strict terms of use for modifications in their home market	Start implementing Machinima tools in games to support Machinima production Provide 'how-to' information and tutorials with games Show tolerance towards application of assets and offer special Machinima licenses	Provide support on online customer community platforms	Enable commercial success by reaching agreements with user entrepreneurs about the commercial use of (their property as) complementary assets Organize competitions among consumers to produce films with their games
	Lead user community		Educate new talent Splits into several sub-communities, Machinima.com as the center of community decreases in importance due to a variety of new platforms and outlets		Joint production on a project basis, sponsored by platforms such as Machinma.com

Table 4: Stakeholder analysis considering the sequence of activities and industries

IMPLICATIONS FOR RESEARCH AND MANAGEMENT

Our findings contribute to the literatures on user entrepreneurship and (remotely) strategy, in that we motivate future research on entrants keeping a low profile to overcome entry barriers. Four contributions stand out.

First, a central finding of our study relates to user entrepreneurship. Industry entry by user entrepreneurs could be observed to occur in two phases (see Fig. 1 and Table 1). This demonstrated that entrants could outmaneuver entry barriers by operating under the radar of incumbent firms before commercializing. Although they were already distributing film, users did not appear as commercial players. Once they had gained a foothold through their audience, users began to commercialize. Their two-phase entry into the animation industry avoided apparent conflict over IP rights by first relying on informal copyright practices (see Lee, 2008), and also avoided large investments in established distribution channels by using the Internet.

Second, our findings contribute to research in entrepreneurship and effectuation more specifically. Rooster Teeth produced Machinima in Halo 1–3, Sims 2, F.E.A.R., and Shadowrun. The team' skill in attracting a community of over half a million subscribing viewers and fans resulted in several sources of revenue through their franchises: DVD sales, sponsorship, merchandising with various items, as well as the production of a comic series that features the Rooster Teeth crew as main characters. The comic was available for free through their website and sold in print. The ventures grew around Machinima but developed Rooster Teeth into a web-based entertainment group with artistic products, such as the comic and the communication contents. Rooster Teeth's team learned by experimentally setting up a shop around its brand and around specific characters from its Machinima series. Rooster Teeth's strategy was driven by its creative use of complementary assets. Using another video game led to an all-new series. According to our interviewees, using the team's gaming experience to relate to other gamers in the audience creates customer loyalty and sparks creativity, in terms of customer-generated content on the community site as well as within the Rooster Teeth team (BB, GR). A traditional sitcom around video gaming could fit with the Rooster Teeth's strategy just as well as more Machinima. The point here is not that user entrepreneurs' creativity is unlimited but that their accumulated experience in the production of Machinima, and the gaming culture that the team shares with its audience, are sources of new ideas that can give rise to new opportunities and ultimately to commercialization. User entrepreneurs search for new applications for assets and skills (their givens) rather than focusing on a predefined target by discarding alternatives, a process that Sarasvathy (2001) terms "effectuation." This process could also be observed in other firms in our sample.

This finding confirms the notion that user innovation rests on the advantage provided by sticky knowledge (von Hippel, 1994) and users' "local" perception of their own and their community's needs. However, it also shows that user entrepreneurs effectively leverage community knowledge for entry into new industries. Studying the user entrepreneur's role in the context of fellow users and consumers echoes the recent call by Rindova et al. (2009) to study entrepreneurial activity more deeply to "understand the relationships between change intent and the nature of relationships with other social actors" (2009: 480). We contribute to the body of research on effectuation by delineating the characteristics of a process where entrepreneurs enter a new industry in two phases. Future research may help to predict target industries and entrepreneurial activity based on user innovation and the principal of givens in effectual thought (compare Read et al., 2009).

Third, our theory contributes to a differentiated understanding of the user's role in technology diffusion. Users, the Machinimators in our case, have radically changed the use of video games by using them as stage and input for animation production rather than game play. Faulkner and Runde (2009) describe the diffusion of these user-triggered changes in products. Video games contain both technology (the physics engine that creates a basis for the virtual world) and art (the artwork that makes the virtual world visible and allows for the experience of sensations inside the virtual world). The two can be separated, as we demonstrated, and their diffusion follows slightly different paths, given their creators' choices to protect them from exploitation in different contexts. The new context is animation; since the

advent of Machinima in the late 1990s, video games have been used for animation production in various ways. Hence, users diffuse technologies across industry boundaries depending on their assessment of the components of the technology, on their own costs of re-creating it, and on the outcome of the negotiations with the owners of the IP rights to the technology, which may in turn be a question of timing.

Fourth, our study raises a few important questions about atomistic versus collective user entrepreneurship and the logic of selective revealing (Henkel, 2006). While the user community of Machinimators was instrumental in diffusing knowledge about the application of video games to animation production, process and business knowledge (sequel and storyline development, distribution channels) were revealed to a far lesser extent. Gamer communities that became Machinima communities preceded commercial efforts, a finding that corresponds to other areas of user entrepreneurship (Baldwin et al., 2006, Shah and Tripsas, 2007). What is new, in our case, is that the choice of industry lies in the nature of the users' activity. We were not observing a few renegade firms that decided to venture into an unknown industry; ours were entrepreneurial users who took innovation one step further and commercialized in the industry most akin to the work they had done: animation. The Machinima community does not discuss game play or extensions of games for gaming's sake but develops ever more sophisticated knowledge that combines the use of games and cinematography, as the quote by Friedrich Kirschner (above) illustrates.

Some limitations apply to this study. First, while the Machinima phenomenon is growing rapidly, generalization of the propositions to areas other than games must be tested in future research. Second, two effects were almost impossible to disentangle and could jeopardize the future applicability of the theory to contexts other than virtual environments: the diffusion of broadband access and advances in distribution technology for media content over the Internet. The popularity of Machinima, and thus the commercial viability of the firms studied, might be connected to the general trend of viewing film over the Internet. Finally, Machinima represents a very recent phenomenon that provides scarce data on the survival of our sample firms (none of which can demonstrate a track record of more than ten years). Game companies could eventually cease selective sharing of engines and tools and venture into the motion picture industry themselves, or the audience could turn away from Machinima. However, given the Machinima firms' diverse sources of revenue, their dependence on game companies is limited. Future research needs to follow up on the phenomenon.

5.1. To be continued…

Our analysis covered a number of topics beyond and in addition to the process model, which we cut in favor of length and accessibility (see also Table 3). We thus propose an agenda for future research. Successful industry entry implies a gain in market share by the new entrant at the expense of incumbent firms. To date, Machinima is insignificant compared to the revenues generated by big players in the motion picture industry. Measuring the market share to justify successful entry thus proves impractical. Machinima represents an emerging genre in film today and user entrepreneurs, entering under the radar, created a new niche in the animation industry, fragmenting existing markets. The new niche is characterized by low barriers to entry, which in turn enable further talent to enter the Machinima market. With the continuously advancing graphical capabilities of video games, the full potential of this market might yet to be unleashed. How this industry will evolve over time, and whether it will capture market share or disentangle from the conventional media markets, remains to be seen. Machinima represents a fertile ground for studying the strategic impact of industry entry and market fragmentation on incumbents as well as new entrants.

The cases demonstrate that learning can occur in phase one, prior to full entry. Hence, important industry entry barriers do not deter user entry and may need reconsideration in terms of their effectiveness for this type of activity (Porter, 1980; Lieberman, 1987). If de novo entrants can innovate without entering an industry commercially (users learn by experimenting with products and processes and from peers), learning curve effects could in fact moderate other entry barriers. Opportunity costs of users are rather low, so could offset entry costs, while they rise with commercialization when the user entrepreneur engages in firm foundation. This eventually leads to the question of which entry barriers remain intact during phase one, industry entry, and which during phase two, entry to the commercial marketplace—a question we defer to future research.

Interaction with their customers helped user entrepreneurs to improve their products and correct flaws. Posting Machinima on sites with social software features (such as commenting and rating) allowed user entrepreneurs to read viewers' comments within minutes.[17] Rooster Teeth encouraged short feedback cycles by establishing a dual release structure granting sponsors early access to new episodes. Sponsors' feedback thus enabled the production team to correct flaws prior to the public rollout. Consumer motivation to engage in such activities, as well as the impact of social software features to support these processes, need close examination in order to implement value-adding IT solutions. The nature of this interaction contributes to the continuous blurring of boundaries between the traditionally separated media of film, Internet, and video gaming.

Firms in one industry increasingly witness how freely or restrictively shared tools and assets can be used in another industry. This can be both beneficial and detrimental to the core business and thus needs close monitoring. In general, our sample suggests that positive effects dominate for game industry incumbents. Ill-considered handling of IP, whether in the form of a too lenient or too strict position on exploitation of complementary assets in users' target industries, can negatively affect the core business. Game companies might want to take a clear stance and encourage or discourage users from certain practices from the point of product launch. While game companies did this rigorously for their home industry, they largely ignored setting up directives for user entrepreneurs' target industries. These policies can define both the kind of exploitation and possible target industries.

Laws differ about the fair use of IP across countries and create uncertainties for users, as our study confirmed. Users can be frustrated with EULAs and select games based on their availability and flexibility of use. EULAs do not necessarily need to be rewritten, nor will individual contracts and agreements resolve the problem in the long run. Amendments can give certain rights back to users, as happened on Blizzard's World of Warcraft site where the company made a 180-degree policy change in 2007 and started to share assets for certain uses selectively. Microsoft issued a similar license in the same year.[18] A recent analysis by Lee (2008) documents the rise of informal copyright practices for user-generated content. This "warming to unauthorized use" (Lee, 2008) includes commercial applications and may, given appropriate and balanced policy (Roquilly, 2010), influence the behavior of both users and entrepreneurs. However, this is an emerging legal practice and the firms in our sample avoided building their ventures on the unauthorized use of copyright work.

Innovation policy needs to take into account the way protected assets can be used across industries. Crucially, the transaction costs to create legal agreements should be lowered so that rights holders have incentives to enter negotiations with prospective entrepreneurs. Today, these incentives frequently point the other way and lead to wholesale decline. Recent Internet technology, such as video compression and distribution, allows new talent to create entertainment products on a broad scale (Eliashberg et al., 2006). Encouraging innovation in this domain calls for more flexible and informal copyright agreements, since the current uncertainty may deter user innovators and user entrepreneurs.

[17] Season five of Red vs. Blue averaged 1,163 comments per episode, the first 50 posts usually arriving within 30 min of the episode's release.

[18] www.worldofwarcraft.com/community/Machinima/letter.html. An article in Wired discussed the license change by Microsoft: www.wired.com/culture/art/news/2007/09/Machinimalicenses.

APPENDIX

1 Machinimator Community
- 1.1 Personal Relationships
- 1.2 Peer Learning
- 1.3 Prices / Promotion

2 Domain Knowledge
- 2.1 Education
- 2.2 Work Experience
- 2.3 Prior Film Production
- 2.4 Hobby Experience

3 Preference Towards Revealing
- 3.1 Engines
- 3.2 Assets / Artwork
- 3.3 Filming Features

4 Complementary Assets
- 4.1 Gaming Experience
- 4.2 Social Software Platforms
- 4.3 Games / Engines
- 4.5 Machinima Tools
- 4.6 Assets / Artwork

5 Tolerance Towards Exploitation
- 5.1 Licensing / Copyrights
- 5.2 Contracts
- 5.3 Support / Forums
- 5.4 Sponsoring / Festivals

6 Learning
- 6.1 Beta- / Pre-Releases
- 6.2 Viewer Feedback
- 6.3 Accumulation of Experiences

7 Horizontal User Innovation
- 7.1 Product (UI-Story)
- 7.2 Published Movies
- 7.3 Diffusion (Distribution)
- 7.4 Related Products
- 7.5 Commercialization

Table A1. Coding scheme

Interview date	Initials	Name	First Name	Company	Country	Profession	Duration [min]	Face to face
Sept. '06	J.B.	Baur	Jonas	Grundy UFA	D	TV Producer	120	x
Sept. '06	K.K.	Kuitenbrouwer	Klaas	Mediamatic	NL		45	x
Sept. '06	D.v.G.	van Gils	Daniel	KmerBlauwLicht	NL		30	x
11.01.07	A.B.	Bailey	Anthony	Amazon	UK		82	
28.11.06	D.R.	Riedel	David	Meowan	D	Student	37	
12.12.06	H.H.	Hancock	Hugh	Strange Company	UK	User-Entrepreneur	35	
28.11.06	J.H.	Hielscher	Jonas		CH	Student	53	
13.12.06	M.J.	Jahrmann	Margarete	HGKZ	CH	Professor	49	x
05.12.06	F.K.	Kirschner	Friedrich		D	User-Entrepreneur	73	
02.03.07	C.B.	Burke	Chris	Bong + Dern Productions	USA	User-Entrepreneur	103	
23.03.07	A.C.	Corcoran	Andrew	EA Europe	CH	VP Strategic Planning	78	x
28.03.07	P.D.	DeBevoise	Philip	Machinima.com	USA	CEO	40	
15.03.07	M.K.	Kelland	Matt	Short Fuze	UK	User-Entrepreneur	42	
13.03.07	H.L.	Lowood	Henry	Stanford University	USA	Lecturer	53	
01.03.07	I.M.	Moon	Ingrid		USA	Consultant	60	
28.02.07	F.v.L.	von Lohmann	Fred	Electronic Frontier Foundation	USA	Lawyer	46	
03.04.07	B.B.	Burns	Burnie	Rooster Teeth Productions	USA	User-Entrepreneur	52	x
27.03.07	F.D.	Dellario	Frank	ILL Clan / ESC	USA	User-Entrepreneur	85	x
04.04.07	M.H.	Hullum	Matt	Rooster Teeth Productions	USA	User-Entrepreneur	29	x
30.03.07	A.K.	Kang	Anna	Fountainhead Entertainment	USA	User-Entrepreneur	51	x
31.03.07	P.M.	Marino	Paul	Academy of Machinima Arts & Sciences	USA	User-Entrepreneur	63	x
05.04.07	J.P.	Paffendorf	Jerry	Electric Sheep Company	USA	Chief Futurist	40	x
03.04.07	G.R.	Ramsey	Georff	Rooster Teeth Productions	USA	User-Entrepreneur	36	x
03.04.07	J.S.	Saldaña	Jason	Rooster Teeth Productions	USA	User-Entrepreneur	30	x
27.03.07	W.S.	Selzer	Wendy	Brooklyn Law School	USA	Lawyer	65	x
02.04.07	G.S.	Sorola	Gus	Rooster Teeth Productions	USA	User-Entrepreneur	28	x
						Count	**25**	**14**

Table A2. Interview partners

INITIATING PRIVATE-COLLECTIVE INNOVATION:

THE FRAGILITY OF KNOWLEDGE SHARING

Simon Gächter
Georg von Krogh
Stefan Haefliger

University of Nottingham
CESifo and IZA

Chair of Strategic Management and Innovation
Department of Management, Technology, and Economics (MTEC)
ETH Zürich, Switzerland

Published in Research Policy vol. 39 issue 7 2010 pages 893-906

Incentives to innovate are a central element of innovation theory. In the private-investment model, innovators privately fund innovation and then use intellectual property protection mechanisms to appropriate returns from these investments. In the collective-action model, public subsidy funds public goods innovations, characterized by non-rivalry and non-exclusivity in using these innovations. Recently, these models have been compounded in the private-collective innovation model where innovators privately fund public goods innovations. Private-collective innovation is illustrated in the case of open source software development. This paper contributes to the work on this model by investigating incentives that motivate innovators to share their knowledge in an initial situation, before there is a community to support the innovation process. We use game theory to predict knowledge-sharing behavior in private-collective innovation, and test these predictions in a laboratory setting. The results show that knowledge sharing is a coordination game with multiple equilibria, reflecting the fragility of knowledge sharing between innovators with conflicting interests. The experimental results demonstrate important asymmetries in the fragility of knowledge sharing and, in some situations, more knowledge sharing than theoretically predicted. A behavioral analysis suggests that knowledge sharing in private-collective innovation is not only affected by material incentives, but also by social preferences such as fairness. The results offer general insights into the relationship between incentives and knowledge sharing and contribute to a better understanding of the initiation of private-collective innovation.

INTRODUCTION

Explaining why and under what conditions innovation happens is a major task for innovation scholars. Software like Linux, Apache, or Firefox, brought the open source software (OSS) movement from obscurity into the public domain and in the process generated much debate among academics and practitioners. OSS is to a large part created by software developers who voluntarily and freely share their knowledge, in what von Hippel and von Krogh (2003) have termed the "private-collective innovation model."[19] This model does not require innovations to be published or released under an open source license, but the innovator forfeits any means of appropriating exclusive returns from selling the innovation. Innovators following the private-collective model of innovation only retain partial ownership of the intellectual property they create, giving up their right to sell and control the use of the innovation after publication or release (see Boldrin and Levine, 2001, and Bessen and Maskin, 2009, for a critical discussion of intellectual property and patents). This model prompted a number of empirical studies on the motivation of OSS developers (e.g. Lakhani and Wolf, 2005; Hertel et al. 2003). On the one hand, research found that participation in a community of software developers could explain contributions to OSS. The community rewards developers who innovate and behave consistently with the community's social norms, by providing new software, discussion platforms to exchange ideas, assistance in improving software, employment opportunities, or public recognition of their effort. These rewards offset the developers' contribution costs (Hertel et al. 2003; Roberts et al. 2006). On the other hand, these studies also revealed a puzzle: what motivates the initiation of private-collective innovation before a community has been established and such social norms have emerged? None of the previous studies could shed light on this question because they focused on contributions to running software development projects. Yet, the answer to this question is crucial. This paper aims to contribute to a better understanding why the first innovator shares knowledge. If we can specify conditions under which the innovation processes emerge, we then may assess the general utility of private-collective innovation beyond established OSS communities.

The essence of private-collective innovation can be illustrated in a simple dyadic relationship: one innovator (firm, entrepreneur, leader, OSS developer) shares knowledge with (at least) one other innovator. In general, this knowledge can be tacit or explicit (Nonaka, 1994), but explicit knowledge (e.g. articles, comments, ideas, engineering plans, design drawings, formulas, algorithms, procedures, software, etc.) can more easily be shared and become non-exclusive and non-rival (Arrow, 1984). During the initiation of private-collective innovation, with no access to public funding, employment contracts or social norms, any innovator can choose to share his or her explicit knowledge as a public good (using, for example, an open source license), or keep it private and use secrecy or intellectual property rights to appropriate private (exclusive) returns from the innovation for example by licensing it to a third party. Under many circumstances, the innovator's incentive to conceal rather than to share knowledge is strong. For example, if an innovator has an idea for a new product that shows great market potential if licensed as a commercial product (rather than using an open source license), the incentive to defect from private-collective innovation can be strong. Several scholars have argued that knowledge sharing is often hampered by such massive conflicts of interest (e.g. Huber, 1982; Michailova and Husted, 2003; Cabrera and Cabrera, 2002; Osterloh and Frey, 2000), and will be fragile at the initiation of private-collective innovation. Authors have suggested that a cost-benefit analysis could shed more light on how different interests and incentives influence innovators' propensity to share knowledge (Foss and Mahnke, 2003: 78–79). In short, two questions stand out in the initiation of private-collective innovation: first, how fragile is knowledge sharing?; and second, what cost and benefit incentives are sufficient to induce sharing rather than concealing knowledge in the initiation of private-collective innovation?

Conducting research on the motivation for initiating private-collective innovation is challenging. Once innovators have shared knowledge and set

[19] Not all OSS is available as a public good. However, if the software is not available as a public good and is, say, created and used by a single developer, that devel- oper follows the private, rather than the private-collective model of innovation. By definition, private-collective innovation entails the publication or release of innovation (von Hippel and von Krogh, 2006). For a definition of open source turn to: http://www.opensource.org/licenses.

innovation in motion, they can be identified by the field researcher. However, since the decision to share knowledge is private, it is difficult if not impossible to study knowledge sharing in the field, and especially difficult to identify innovators who decided not to share their knowledge and their reasons for doing so (see Nonnecke and Preece, 2000). To circumvent these difficulties, we model the incentive structure behind knowledge sharing and make use of an experimental method to understand the behavioral consequences of these incentives. The paper contributes to the literature on private-collective innovation by questioning what incentives motivate innovators to share their knowledge in an initial situation, before a community and social norms are available to sustain private-collective innovation.

In section two of this paper, we review existing research, and discuss the role of incentives in private-collective innovation. We develop a game theoretic model of knowledge sharing at the initiation of private-collective innovation. We show that knowledge sharing is a coordination game with multiple equilibria. Since there are many outcomes of knowledge sharing, we develop an experimental research design to identify behavioral strategies, given different incentives. The third section explains our research design and methods, and the fourth section presents the results. We find there are important asymmetries in knowledge sharing and, in some situations, much more knowledge sharing than theoretically predicted. A behavioral analysis suggests that knowledge sharing is affected not only by material incentives, but also by social preferences, such as fairness. In Section 5 we discuss our findings and conclude.

INCENTIVES FOR KNOWLEDGE SHARING IN PRIVATE-COLLECTIVE INNOVATION

Three models have specified incentives that give rise to innovation in society and economy: the private-investment model (Demsetz, 1967; Arrow, 1984), where the innovation remains a private good for the innovator who retains the rights to consume it, sell it, or provide access to it for third parties for a fee; the collective-action model, which relies on collective or public subsidy for public goods innovations (e.g. Olson, 1965; Stephan, 1996; Dasgupta and David, 1994) and where innovators relinquish control of knowledge or other assets they have developed and make them a public good (Hargrave and Van de Ven, 2006; Garud et al., 2002); and the private-collective innovation model (Von Hippel and von Krogh, 2003), where innovators fund public good innovations voluntarily and privately. OSS is often discussed as an exemplar of the latter model. In thousands of OSS projects in existence today, ranging from the operating system GNU Linux to the Firefox browser, individuals, research teams, universities, firms, and governments give their money, limited time, and talent to create software free for all to inspect, download, use, modify, and freely redistribute to others in a modified or unmodified form. The term "open source software" refers to copyright licenses, such as the GNU General Public License (GPL), that simultaneously guarantee access to the source code to users of the software and keep the innovation available as a public good once it has been shared with users (see discussion of this point in Lerner and Tirole, 2002; O'Mahony, 2003; Osterloh and Rota, 2007). Although open source licenses vary in restrictiveness[20] and in their tolerance for integration with proprietary software, the GNU General Public License is the most commonly used. To date it has been used by 124,000 out of 170,000 projects listed on sourceforge.com, and it forms the backdrop of this paper.

The private-collective model proposes that the efforts and participation of innovators in a community create incentives to innovate. A number of empirical studies have provided evidence for this proposition. Research has uncovered community-related incentives. These include: the application and testing of the software by many users and developers (Raymond, 1998; Lakhani and Wolf, 2005; Shah, 2006); the adaptation of OSS to solve specific technical problems on several developers' computers (Franke and von Hippel, 2003); the learning that takes place when users share knowledge and jointly write OSS (Kuk, 2006); the individual software developers' creation of software modules that can be combined into a whole software product by the work of many (Baldwin and Clark, 2006); the reputation that individuals achieve among peer developers in the community (Roberts et al., 2006); the developers' obligation to help fellow software users solve problems installing programs on their computers (Lakhani and von Hippel, 2003); and the developers' identification with a specific community (Hertel et al., 2003). In their review of research on OSS, Bergquist and Ljungberg (2001) suggested that a cornerstone of OSS developer communities is the way they evolve strong social norms of reciprocity that enable them to operate as "gift economies." When software developers receive "gifts," in terms of advice, acknowledgment, or software from others they feel an "obligation" to reciprocate by giving new or improved software, advice, and tips.[21]

The private-collective model of incentives to innovate presupposes an active community of innovators, and until now has not explicitly covered the initiation of innovation (von Hippel and von Krogh, 2003; 2006). It has always been understood as "routine collective action" (Useem, 1998), where innovators build on already existing technology to create new and useful products. However, incentives during initiation differ substantially from incentives once the community is established and working routinely. According to Elster (1986), the initiation of collective contribu-

[20] Differences in restrictiveness apply to the use of OSS in conjunction with pro- prietary software. The GNU General Public License mandates a so-called "copyleft" provision stating that all other software derived from or used together with software licensed under the GPL adheres to the same license terms. Other OSS licenses, such as the BSD License, are less restrictive or, in other words, allow copies and deriva- tives to be used in proprietary software as long as the authors are acknowledged (see, e.g., http://www.openbsd.org/policy.html).

[21] Lately, studies have also found private-collective innovation incentives in other fields than software, including product development and cultural goods (e.g., de Vries et al., 2006; Jeppesen and Frederiksen, 2006).

tions to a public good requires a higher incentive level than that required to sustain them.

Consider four examples of research on OSS development that highlight the evolving nature of incentives as communities grow and prevail. First, an initial gift (Bergquist and Ljungberg, 2001) must be exchanged before a social norm of reciprocity can be established in the community. OSS developers may only feel gratitude and an obligation to return a gift if they have first received useful software, appreciation, or advice. Second, when a developer releases a first working version of OSS to the public, other developers who find the software product useful will join the development effort and build a community (Raymond, 1998; Lerner and Tirole, 2002). The role of working software (a public good) in generating development activity has been confirmed by statistics on new developers joining the Freenet open source peer-to-peer project (von Krogh et al. 2003). Third, Baldwin and Clark (2006) showed that the modularity of software architecture is positively related to the options available to OSS developers in terms of exchanging valuable work. The value of these options, and hence the incentive to contribute, hinges on a modular structure of existing software architecture already created by many developers. Fourth, Shah (2006) found that long-term OSS developers take on mundane tasks in a community, such as giving advice to newcomers or maintaining mailing lists. This work reaches beyond the developers' immediate, individual goals of satisfying their technical needs. Thus the motivation for contributing to the community changes over time in an OSS project.

The evidence that incentives change as communities emerge and grow suggests that we cannot infer characteristics of initial knowledge-sharing situations from our observations of established private-collective innovation; nor can we make assumptions about levels of incentive in such situations. Therefore, in order to capture the initiation of private-collective innovation before the establishment of a community and the emergence of social norms, we introduce two rather weak assumptions from von Hippel and von Krogh (2003; 2006). These allow us to model the basic structure of knowledge sharing (sequentiality and asymmetry in conflicts of interest) and can be illustrated by a sequential two-person game that models an initial knowledge-sharing situation:

1. Knowledge sharing enhances value for the party that receives the knowledge.
2. Mutual knowledge sharing makes knowledge public and precludes the exclusivity of received knowledge for one's own private financial benefit. By implication, a necessary (but not sufficient) condition for exclusive appropriation is unilateral knowledge sharing.

Assumption 1 states that knowledge sharing is value enhancing (net of adoption costs of the new knowledge) for the innovator receiving the knowledge. This is an innocuous assumption, because in case it does not hold, knowledge sharing is not really of economic interest. Value enhancement occurs if one party shares his or her knowledge, and is further enhanced if there is mutual knowledge sharing (von Hippel and von Krogh, 2003). Assumption 1 does not state whether or not the act of sharing knowledge is costly for the innovator. We will discuss this later. Note also that knowledge is distinct from other resources, such as land or money. Sharing it does not necessarily diminish the rewards it will bring for the innovator who first held it. For example, in OSS, the software developer who shares his source code with another developer will still have a copy that can be used to solve technical problems on his own machine. By "innovators" we mean both individuals and firms. Recent research provides evidence that firms often reveal knowledge freely and extensively, including software code released under an open source license and developed in-house (e.g. Henkel, 2006; Stuermer et al., 2009).

Assumption 2 says that mutual knowledge sharing rules out the exclusive appropriation of received knowledge for one's own private financial benefit (von Hippel and von Krogh, 2003). This is because mutual knowledge sharing makes knowledge a public good and, by definition, the returns on a public good cannot be exclusive (Arrow, 1984).[22] Appropriation refers to the capacity of the knowledge holder to receive a return equal to the value created by the knowledge, for example, by keeping it secret or by protecting it

[22] An example would be publishing a software (and source code) on the Internet, with everybody being allowed to use the software for free. In this case a private market is not viable any longer and hence exclusivity is ruled out. Assumption 2 does not preclude products, services, and business models that build on the public knowledge by combining it with complementary assets and skills, thus partial, or non-exclusive, appropriation.

through intellectual property rights (Arrow, 1984; Teece, 1986; Grant, 1996). An individual who combines her own concealed knowledge with knowledge received from others may receive a higher return than the returns from knowledge sharing. Sharing knowledge entails opportunity costs of giving up exclusivity that must be added to the possible out-of-pocket costs of sharing.

Assumption 2 is important because it allows us to model conflicts of interest as a possible consequence of unilateral knowledge sharing. This asymmetry captures the real-life differences between innovators' prior knowledge and their opportunities. For example, a programmer may face a technical problem. She combines her software with a technical solution provided by others, and in doing so learns to solve the problem more efficiently. If, thanks to this combined knowledge, the programmer can create software that allows her to sell a software product on the market, she might choose to release a binary version of the software only and license the software to third parties in return for a fee. However, if the software is protected by an OSS license, the programmer's ability to sell the software is limited by others' rights to do the same or give the software away for free, in other words, exclusive appropriation becomes impossible.[23] As we mentioned earlier, most open source licenses guarantee the rights of current and future users to freely download, inspect, modify, and release modified and unmodified versions of the source code (not the binary version) to third parties. Open source licenses thus lower the opportunity costs of sharing, suggesting a model structure that allows us to observe the sensitivity of mutual sharing to varying levels of conflicts of interest.

Knowledge is accumulative; it enables learning and can provide complementary benefits to either of the innovators involved. In the absence of the sort of hierarchy, incentive structure, employment contracts and other contextual factors found in firms, innovators act according to social preferences, such as fairness, efficiency-seeking, and reciprocity. To uncover what motivates the initiation of private-collective innovation, we now turn to some basic incentives for knowledge sharing.

2.1. The knowledge-sharing game

The generic properties of the conflicts of interest in knowledge sharing that follow from our two assumptions can be illustrated in a sequential dyadic relationship. An example of sequential knowledge sharing is an individual's decision to contribute software to an existing open source project and add his or her own solution to what has already been published.

Figure 1 illustrates our "knowledge sharing games." For simplicity, we call the two innovators leader (L) and follower (F). Both innovators have the choice of sharing (s) or concealing (c) knowledge. In the knowledge-sharing game in Figure 1, L moves first and decides whether to share or conceal his knowledge. F is informed of L's choice and decides whether to share or conceal his or her knowledge. Note that the model allows F to decide whether or not to share knowledge, even if L has decided to conceal.[24] After F's choice, the game ends and payoffs are realized: bi (i = L,F) denotes a base payoff where vi is the value enhancement through sharing knowledge, ai is the exclusivity payoff, and k denotes the expenses for sharing explicit knowledge. These payoffs are derived from the assumptions and explained in detail below.

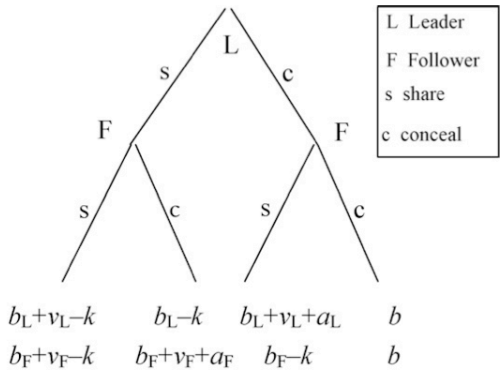


Figure 3: The knowledge-sharing game

[23] Other business models derived from the use of publicly available software, e.g., in combination with proprietary software, can still create conflicts of interest for the receiver. Henkel (2006) showed that mutual revelation can be compatible with private benefits in the example of embedded Linux. Further private benefits can include intrinsic motivation (Osterloh and Frey, 2000; Ryan and Deci, 2000).

[24] In our example, this would correspond to adding a solution to an existing project even if the project leader has not (yet) contributed code. The famous example is Linus Torvalds' discussion of a new operating system before publishing code (Moon and Sproull, 2000; Moody, 2001). Fora or mailing lists may provide a social context independent of the initial sharing of code.

Before analyzing the incentive structure and inherent conflicts of interest in the knowledge-sharing game, payoffs must be specified. Each actor receives some basic net benefit bi ≥ 0, i = L, F, even if he or she only retains knowledge. An example would be programmers who produce software that is useful to them without receiving any knowledge from another programmer. According to Assumption 1, net value is enhanced for the knowledge recipient. In Figure 1, this is reflected in payoffs vi > 0, i = L, F, to player i if player j ≠ i shares his or her knowledge. For instance, if L shares his or her knowledge, then, irrespective of F's choice, F receives payoff vF in addition to his or her basic payoff bF. If F shares his or her knowledge, then L's base payoff bL is augmented by payoff vL. If player j conceals, then vi=0, i = L, F.

Assumption 2 says that knowledge (a public good created by mutual sharing) cannot be turned into an exclusive benefit. Therefore, a necessary (but not sufficient) condition of turning received knowledge (in combination with one's own knowledge) into an exclusive benefit is that knowledge sharing is unilateral. Only the innovator who can combine his or her own knowledge with knowledge obtained, and does not share his or her own knowledge, can possibly enjoy the benefits from exclusive appropriation, denoted by ai ≥ 0, i = L, F. Therefore, if L shares and F conceals, F can exclusively appropriate aF. Likewise, L only appropriates aL, if he conceals and F shares his or her knowledge. Notice that the payoff connected to exclusivity can be zero (which is why asymmetric sharing is only a necessary, not a sufficient, condition). This is the case, for instance, if exclusive appropriation of shared software is prevented by an Open Source license (see O'Mahony, 2003, for a discussion of various protective measures for OSS). Exclusivity is the crucial variable in the subsequent experiment, as will become clear in the game-theoretic analysis. Intuitively, exclusivity constitutes the opportunity costs of sharing knowledge, because benefits from exclusive appropriation are foregone if knowledge is shared.

One might argue that, in real life, sharing explicit knowledge in innovation is costly (for example shipping documents or traveling to meet people). The cost of knowledge sharing is modeled with the variable k.[25] These costs are "out-of-pocket expenses," distinct from the opportunity costs that are created by exclusivity (private returns from the exclusive appropriation of knowledge). If knowledge sharing is costless, k = 0. OSS development is characterized by this condition. Von Hippel and von Krogh (2003) and Kogut and Metiu (2001) argue that the internet makes the sharing of knowledge between programmers a near costless activity. This is in contrast to the high out-of-pocket expenses needed to share or replicate many other technologies across sites, such as manufacturing processes, physical product prototypes, or chemical compounds (Kogut and Zander, 1992, and Teece, 1977, discuss many costs involved, such as local adaptation of the technology to the manufacturing site simply to make it run).

2.2. Theoretical predictions

We are now ready for a game-theoretic analysis of the conflicts of interest inherent in this knowledge-sharing game. First, we assume that players are rational and purely self-interested. Second, we predict the outcomes considering fairness as a social preference, building on the model by Fehr and Schmidt (1999). Fairness considerations play an important role in bilateral exchanges and apply to a broad range of economic situations (Fehr and Gächter, 2000).

Observe the implications of positive out-of-pocket costs k > 0. In case vi > k > 0, the sequential knowledge-sharing game is a sequential prisoner's dilemma game, where the only equilibrium is mutual concealment. This holds irrespective of the exclusivity ai.[26] Therefore, in the present model, knowledge sharing by self-interested innovators is not possible if k > 0 (of course, mutual concealment is inefficient). Since both players want to conceal in the sequential prisoner's dilemma, there is no genuine conflict of interest either. Therefore we confine our attention to the consequences of opportunity costs in sequential knowledge sharing if k = 0 for both players.[27]

[25] We assume that (i) k is the same for both players because market prices for shipping costs or access to the internet are the same for everyone, and (ii) k < vi , i = L, F (the costs of knowledge sharing are smaller than the value created).

[26] If the leader and the follower decide simultaneously and k > 0, the knowledge- sharing game becomes a prisoner's dilemma.

[27] If sharing knowledge is actually beneficial for the sharer, i.e., if the sharer would be paid for the act of sharing, k < 0. The implications are that, if abs(k) > aF , F will always share whether L conceals or shares. In this case the result is that mutual sharing is an equilibrium if abs(k) > aL .

The following proposition summarizes the knowledge-sharing game:

Proposition 1: Knowledge sharing based on rationality and self-interest:

(i) Mutual concealment is always an equilibrium outcome.

(ii) A necessary condition for mutual sharing as an equilibrium outcome is that $aF = 0$. A necessary and sufficient condition for sharing in all subgames is that exclusivity payoffs are zero for both players ($aL = aF = 0$).

(iii) In all constellations of exclusivity payoffs ($aL \geq 0$)x($aF \geq 0$), there always exist equilibria with unilateral knowledge sharing (in addition to mutual concealment).

The first basic message of the proposition is that the knowledge-sharing game has multiple equilibria. Appendix A contains the complete list of these. Proposition 1(i) makes it clear that in addition to possible unilateral and mutual knowledge-sharing equilibrium outcomes, mutual concealment is always an equilibrium. The intuition for this result is that, under $k = 0$, F will, in both subgames where she has to make a decision, either be indifferent to sharing and concealment (in the subgame after L chooses c, or in the subgame after L chooses s and $aF = 0$) or be better off by concealing (in the subgame after s and if $aF > 0$). Thus, concealment is always a best response for F. It then follows that it is best for L to choose c as well, since he is indifferent to sharing and concealment in this case.

Proposition 1(ii) says that mutual sharing will only occur as an equilibrium outcome if F cannot gain exclusivity payoffs. Since F is indifferent to both sharing and concealment in this case, F may resolve her indifference by sharing. Given that F shares, L may also share, provided $aL = 0$. In this case L is equally indifferent to sharing and concealment (L's payoff is $bL+vL$ anyway). As soon as $aF > 0$, F will conceal, if L shares and mutual sharing can no longer be an equilibrium. Thus, $aF = 0$ is a necessary condition for mutual sharing; $aL = aF = 0$ is necessary and sufficient.

The rationale for proposition 1(iii) is that in cases of indifference to both sharing and concealment, both choices are a best response. That is, when matched with the other innovator's best response of sharing, concealment can be a best response, and vice versa.

Proposition 1 shows that genuine conflicts of interest, where one innovator wants to conceal and the other share, can only arise if there are positive opportunity costs to sharing knowledge for one innovator alone. There is no conflict of interest in the mutual knowledge-sharing (-concealment) equilibria, because it is in both innovators' interest to share (conceal) their knowledge.

A particularly interesting situation is the one where neither innovator has opportunity costs of sharing (i.e., $aF = aL = 0$). As von Hippel and von Krogh (2003) argue, this situation is characteristic of many open source licensed projects. The analysis shows that the resulting open source knowledge-sharing game is no prisoner's dilemma, but a game with multiple equilibria with different efficiency consequences. Mutual knowledge sharing can occur simply because innovators are indifferent to both sharing and concealment and are prepared to resolve their indifference by sharing. It is exactly this indifference that allows for mutual sharing among self-interested innovators.[28] However, by the same token, mutual concealment is also an equilibrium, albeit an inefficient one.

Proposition 1 is crucial for understanding the fragility of knowledge sharing in the initiation of private-collective innovation, because it highlights the structure of the conflicts of interest inherent in knowledge sharing. The dual findings that (i) mutual concealment is always an equilibrium outcome, and (ii) that only equilibria with unilateral knowledge sharing or mutual concealment exist as soon as $aF > 0$, gives a theoretical meaning to the "fragility of knowledge sharing." In sections 3 and 4 of this paper, we complement this theoretical result with evidence of the behavior that causes fragile knowledge sharing. Since knowledge sharing is a game with multiple equilibria, it is an open question which equilibrium innovators will play. This is an inherently

[28] An example would be a programmer who develops software for himself. Sharing it does not diminish the utility of the software for him, but may be beneficial for another programmer. So given that he is not worse off by sharing, and somebody is potentially better off, he might easily be prepared to share his software with others.

empirical question that we attend to in the remainder of the paper.[29]

Proposition 1 summarizes a benchmark game-theoretic analysis under the simplifying assumption that innovators are rational and selfish (i.e., they only maximize their own payoffs). However, the research on open source communities we discussed earlier (e.g., Bergquist and Ljungberg, 2001), and research in psychology and experimental economics, has repeatedly revealed that rather than being selfish, many people are equipped with "social preferences." In addition to pecuniary payoffs, people also care about equity, efficiency, and reciprocity (Camerer, 2003, chapter 2). Since knowledge sharing has various payoff and efficiency consequences, social preferences might also matter in the context of private-collective innovation.

One simple model of social preferences is the inequity aversion model of Fehr and Schmidt (1999). According to this model, players' utilities increase in their own material payoff but decrease if there is inequality in bilateral payoff comparisons. For instance, an inequality-averse F will always conceal if L has concealed because sharing only increases L's payoff but not F's and therefore would put F at a payoff disadvantage (Fig 1). Players in the Fehr-Schmidt model might also dislike advantageous inequality, that is, situations in which they earn more than their co-player. For instance, after L has shared, F might consider sharing if he is sufficiently averse to advantageous inequality, that is, if his dislike of the payoff advantage vis-á-vis L (v + aF) outweighs the material gain compared to concealment (the material payoff gain from concealment compared to sharing is aF).

To see this more formally, consider the Fehr-Schmidt utility function of F: $U_F(\pi_F, \pi_L) = \pi_F - \alpha_F \max[\pi_L - \pi_F, 0] - \beta_F \max[\pi_F - \pi_L, 0]$ where $\pi_i, i = F, L$ indicates the material payoffs as they were relevant in the experiment; $\alpha F \geq 0$ measures F's aversion to disadvantageous inequity ($\pi_L - \pi_F < 0$) and βF measures aversion against advantageous inequity ($\pi_F - \pi_L < 0$) ($\alpha F \geq \beta F \geq 0$; $\beta F < 1$). Plugging the relevant payoffs in the utility function shows immediately that an inequity-averse follower will never share if L has concealed.

F might share, however, if L has shared. F's utility from sharing, given that L has shared, is $U_F(s|s) = b + v$. The utility from concealing is $U_F(c|s) = b + v + a_F - \beta_F(b + v + a_F - b)$, that is, F enjoys the material benefit of $b + v + a_F$ and suffers a disutility of $\beta_F(v + a_F)$ from earning $(v + a_F)$ more than L. F will share if and only if $U_F(s|s) \geq U_F(c|s)$ which is the case if $\beta_F \geq a_F/(v + a_F)$. In our experiment v = 20 (see next section). Therefore, our different levels of the exclusivity payoff require the following minimal $\beta^*(a)$ to induce F to share: $\beta^*(0) = 0; \beta^*(10) = 1/3; \beta^*(20) = 1/2$ and $\beta^*(30) = 3/5$. Thus, the higher the benefits from exclusivity the higher has F's aversion against advantageous inequality to be to induce him to share despite giving up exclusivity.

What can we say about L's behavior? If L expects F to share, it will be best for L to share, as L's payoff for sharing will be b + v rather than b if he conceals. If L expects F to conceal, it will be best for L to conceal as well. We summarize the role of inequity aversion in knowledge sharing in our second proposition:

Proposition 2: knowledge sharing under inequity aversion

(i) If followers are even slightly inequity averse they will always conceal if the leader has concealed. If the leader shares the follower will also share, if he or she is sufficiently inequity averse. The degree of inequity aversion needed to induce a follower to share increases in the follower's exclusivity payoff aF.

(ii) The leader will always share if he expects the follower to share and conceal if the follower conceals.

Which parameters for βF are plausible? Based on results from student participants Fehr and Schmidt (1999: 844) assume that 30 percent of people have β = 0; another 30 percent have β = 0.25 and 40 percent have β = 0.6.[30] If we assume that our student participants have similar β-values as those reported by Fehr and Schmidt, and know about the distribution of these values, then we can make the following prediction about

[29] See, e.g., Camerer et al. (1997), Weber et al. (2001) and Camerer (2003), (Chapter 7) for discussions of the difficulties of coordination.

[30] These values should only be seen as a rough approximation, rather than as exact point estimates. See Bellemare et al. (2008) for econometric estimates based on a representative sample.

mutual sharing (remember that L will always share if he expects F to share): If L shares, 70 percent of Fs (with a $\beta > 0$) will share if $aF = 0$ (the remaining 30 percent are indifferent toward the outcome), and 40 percent will share if $aF > 0$.

Apart from inequality aversion, other kinds of social preferences might impact the fragility of knowledge sharing, namely efficiency-seeking and reciprocity. If followers (perhaps in addition to inequality aversion) think concealment is unkind and that leaders who conceal are greedy ("They conceal because they want to appropriate my shared knowledge"), reciprocity predicts that, in this subgame, followers who want to "punish" greedy leaders will conceal.[31]

Efficiency concerns (e.g., Charness and Rabin, 2002; Engelmann and Strobel, 2004) make different predictions if L has concealed knowledge. If F is concerned with efficiency, he might share knowledge because his own payoff is not affected; but L's payoff is increased and efficiency is enhanced.

[31] Choosing c in case L chooses c, is a "punishment" for L, if L expected F to choose s instead, since L's payoff is smaller under c than under s. Reciprocity theories (Rabin, 1993; Dufwenberg and Kirchsteiger, 2004; Falk and Fischbacher, 2006) make similar predictions to inequality aversion in this game and so we do not consider them separately here.

RESEARCH DESIGN AND METHODS

The generic incentive structures in the initiation of private-collective innovation follow from the two basic assumptions we discussed in section 1. As we saw in section 2, it is difficult, if not impossible, to examine possible defection by innovators in field studies of private-collective innovation. Therefore, we will test the behavioral consequences of the model in a laboratory decision scenario that has the same incentive structure as the theoretical model we described earlier. Paying the experimental subjects according to the payoffs of the model ensures that the subjects face the monetary equivalent of the incentives that are assumed in the model (see Smith, 1982, for a comprehensive methodological discussion of this "induced value" technique). In this way, one can observe real economic decisions by human decision makers who face real stakes in a situation that has the same form as the model of knowledge sharing discussed above.

Since (i) the only crucial variables are the exclusivity payoffs, and (ii) none of the theoretical results requires $bL \neq bF$ and/or $vL \neq vF$, we simplify the analysis to focus on variations in a by assuming $bL = bF = b$, and $vL = vF = v$. We also assume that $k = 0$, i.e., out-of-pocket costs are absent. In the experiments, we implemented the extensive form game of Figure 1. We chose the following parameters: $b = 10$ and $v = 20$. We did not change these parameters during the experiment since, theoretically, they are of minor interest. The exclusivity payoffs vary between four levels: $ai \in \{0, 10, 20, 30\}$, $i = L, F$. The rationale for four levels of ai is as follows. Note that for $ai = 0$ or 10, the social optimum (i.e., the sum of payoffs) is always achieved in mutual sharing. If $ai = 20$, both mutual and asymmetric knowledge sharing are socially optimal. If $ai = 30$, the social optimum is that only one player shares knowledge and the other conceals. We included this payoff in the design as well, since there is evidence from experiments that efficiency-seeking is often an important behavioral motive. It is particularly interesting to examine how combinations of aL and aF influence the likelihood of knowledge sharing, and therefore we vary these payoffs systematically, by playing all 16 possible payoff combinations aFxaL ($\{0, 10, 20, 30\}x\{0, 10, 20, 30\}$).

Each participant in the experiments played 16 games—each resulting from aFxaL—once. Subjects were randomly allocated to their roles as leaders or followers. After they had read the instructions (see Appendix B), they made their decisions in each of the 16 games without feedback. We programmed and conducted the experiments in z-Tree (Fischbacher, 2007). For simplicity, subjects saw only the sum of payoffs ($b + v$) or b alone on their screens. Moreover, we told subjects that the only payoffs that would change in each of the 16 games were payoffs for the follower (called y on the screens) that resulted from the share-conceal combination (L chooses s, F chooses c) and the payoff for the leader (called x on the screens) in the conceal-share combination (L chooses c, F chooses s). We did not inform subjects beforehand about the values of x and y. They were told that only these two payoffs would change from period to period. However, subjects did know that there would be 16 periods. The order of x-y combinations was randomly determined but the same for all subjects.[32]

We avoided possibly value-laden content labels for the choices, following common practice in experimental economics. The players were not referred to as "leaders" or "followers," but as "decision maker 1 and 2." Choices were framed neutrally and did not refer to knowledge sharing. Choices were termed left or right (in the game in Figure 1, left corresponds to share and right to conceal).[33] Leaders simply decided whether to choose the left or right option. For followers, we applied the strategy method (Selten, 1967). That

[32] One might object that, despite the lack of feedback, some "virtual learning" (Weber, 2003) might affect all players similarly because they all played the games in the same order. To test whether this procedure had an impact on our results, we ran further experiments (with 51 subjects) where we randomized the sequence of games for each individual (i.e., each individual played the 16 games in a differ- ent sequence). For each of the 16 games we apply tests of proportion to test the null hypothesis that the frequency of leaders' sharing decisions in the experiments, where everyone played the 16 games in a same order, is the same as in the control experiment, where people play the games in different orders. We applied the same test to the followers' sharing decisions after the leader had shared and concealed. In all, we performed 48 tests. We could not reject the null hypothesis (at $p < 0.05$) of equal proportions of sharing decisions in 45 out of 48 tests. We concluded that our results (reported below) are robust to the sequence of play.

[33] The research methodology behind this design choice is to look at the basic incen- tive structure behind knowledge sharing. Future research should address the role of context for the fragility of knowledge sharing.

is, followers had to make a left-right decision whether the leader chose left or right. Neither leaders nor followers received any feedback. The rationale for the strategy method is twofold: first, it allows the observation of F's behavioral reactions to both possible leader choices. This would not be possible if F were restricted to making a decision after he or she has seen a specific leader choice.[34] Second, asking for contingent choices has the added advantage that feedback does not have to be given in each round. This makes decisions between individuals independent, which is advantageous in the statistical analysis of the data. In particular, since with the strategy method without feedback decisions are independent between subjects, we can use the observed sharing frequencies to calculate the expected probability that a randomly matched leader-follower pair (not just the actual matched pair) plays a certain strategy combination, and—of particular interest—the probability of mutual sharing. This would not be feasible without the strategy method and independence of decisions.

To determine payoffs, leaders and followers were matched randomly in each period. However, we did not provide feedback on the 16 games until the end, when subjects received the earnings from each of the 16 games in cash. In each of the games, payoffs were determined according to the decisions of a randomly matched leader-follower pair. During the experiment, payoffs were denoted by points. Finally, we exchanged the accumulated sum of points into Swiss Francs at an exchange rate of one point = 0.04 Swiss Francs.

We conducted the experiments at the Universities of St. Gallen and Zurich in Switzerland, with 228 undergraduates from various fields as experimental subjects. They provided a total of 3616 share-conceal decisions. The experiments lasted 30 minutes and subjects earned on average 15 Swiss Francs (approximately US$ 12.3). Across all games, leaders and followers earned very similar amounts.

[34] There is evidence that in simple coordination games like ours the strategy method does not lead to systematically different responses than ordinary game playing. See Brandts and Charness (2000) for a systematic analysis. They study coor- dination games with and without the strategy method. In the experiments without the strategy method, subjects are simply confronted with the choice of a first mover and then make their decision. Under the strategy method, subjects make decisions for all possible moves of another player. The behavioral results do not differ between the methods.

EXPERIMENTAL RESULTS

Our analysis of the results from the experiments on knowledge sharing in the initiation of private-collective innovation is largely descriptive. Appendix D contains an econometric analysis that corroborates our results statistically.

4.1 Sharing decisions

Since the only parameters in the experiments are combinations of aFxaL, we will present most results as a function of these parameters.[35]

Result 1:

(i) Contingent on the leader sharing, followers share in about 73 percent if aF = 0. If aF > 0 the probability that the follower shares drops dramatically (to less than 30 percent) and decreases further in aF. This holds for all levels of the leader's exclusivity payoff aL.

(ii) When the leader conceals, we find that the probability of followers sharing is on average 45.3 percent across all aFxaL-combinations.

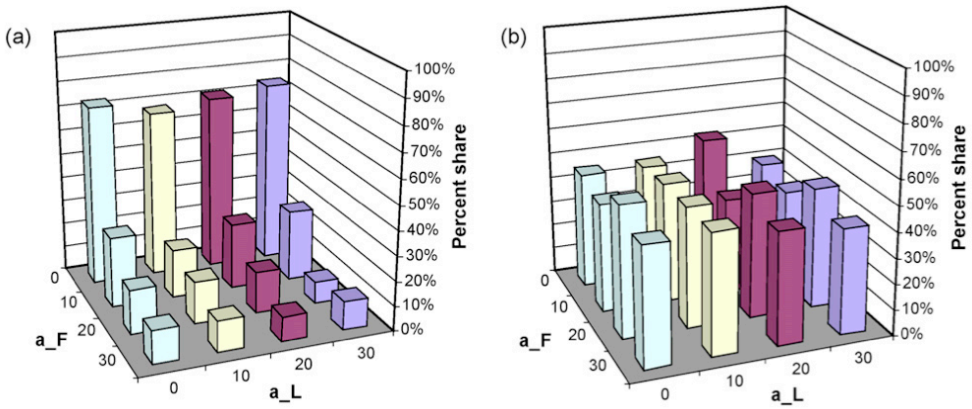


Figure 4: Percentage of followers who share if (a) the leader shares and (b) if the leader conceals

Figures 4a and 4b provide the main support for Result 1. For each of the 16 aFxaL-games, the figures show the frequencies at which F shared in the sub-game after L shared (Figure 4a) and concealed (Figure 4b). A comparison of these figures shows that L's decision strongly affects F's sharing behavior.

Figure 4a illustrates the contingent probability at which Fs are prepared to share knowledge, in case L shares. Recall that under our assumptions of rationality and selfishness, there should not be any knowledge sharing in the event of a positive exclusivity payoff aF. If aF = 0, rational and self-interested followers are indifferent to both sharing and concealment. This implies that behavior is undetermined in this case. To the extent that Fs are inequality averse or care for efficiency, F might be willing to share (Proposition 2). For all levels of aL this was the case for between 67 to 74 percent of followers.

The followers' willingness to share dropped dramatically for aF > 0. This holds for all levels of aL. If aF = 10, the likelihood that F will share was between 20 and 29 percent. If aF = 30 it dropped even further to between 10 to 14 percent.

Figure 4b illustrates the contingent probability at which Fs are prepared to share knowledge in case L conceals. The results show that Fs chose to share in between 37.5 and 53.6 percent of cases. The average over all aFxaL-combinations was 45.3 percent (which is not significantly different from 50 percent, the expected outcome of indifference between sharing and concealing; t-test with individual average sharing rates as observations).

Recall that Proposition 1 predicts that F does not share in case aF > 0, and is indifferent in case aF = 0, whereas Proposition 2 (in combination with plausible parameters) predicts 70 percent sharing if aF = 0 and 40 percent sharing if aF > 0. Thus, the evidence of F sharing after L has shared appears to be consistent with Proposition 2. How-

[35] Appendix C documents the frequency of L's sharing choices for all 16 games, as well as F's choices after L shared and concealed. Since we have information from the full strategy set of our knowledge sharing game, we also document the expected frequencies at which the various strategies are played.

ever, the fact Fs share in almost 50 percent of the cases in which L conceals is inconsistent with inequality aversion but consistent with efficiency considerations.

The next result concerns the probability that the leaders share knowledge in the initiation of private-collective innovation.

Result 2:

The probability that leaders share is affected negatively by both their own and their followers' exclusivity payoffs.

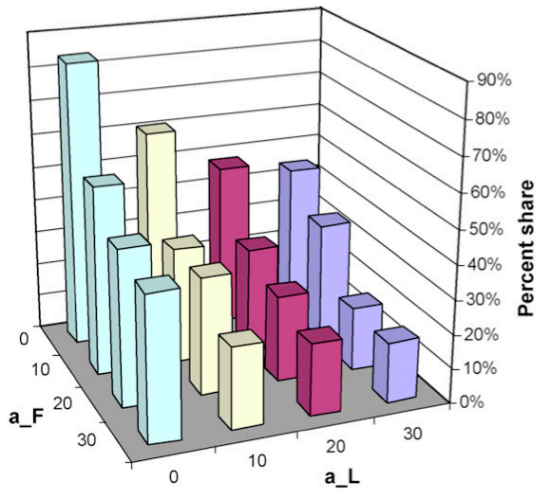


Figure 5: Percentage of leaders who share

Figure 5 is the main support for Result 2, which shows the percentage of cases in which L shared in each of the 16 aFxaL-games.

As Figure 5 shows, the likelihood that L shares knowledge decreases in both aF and aL. In case both exclusivity payoffs are zero, Ls share in 84.2 percent of the cases. The probability that Ls share is lowest if aF = aL = 30 (it equals 17.5 percent). Thus, in the inception of private-collective innovation, Ls take into account not only their own exclusivity payoff aL, but also their followers' (aF).

4.2 The fragility of knowledge sharing

As we have seen, innovators' opportunity costs of knowledge sharing strongly affect the likelihood that mutual knowledge sharing will occur in the initiation of private-collective innovation. We close our empirical analysis by examining the fragility of mutual knowledge sharing, using the possibilities inherent in collecting data with the help of the strategy method with no feedback between rounds. The design of the strategy method (i) provides many independent observations and (ii) allows the observation of strategies, not just realizations of decisions in the game. The likelihood of a certain outcome can be estimated (see Appendix C).

Fragility is operationalized as the change in the expected probability of mutual knowledge sharing, when the opportunity costs of sharing change. Note that the fragility of mutual knowledge sharing is a composite of L's and F's sharing behavior. It is therefore an empirical question whether L's or F's behavior is more important for the fragility of mutual knowledge sharing. The result is as follows:

Result 3:

Knowledge sharing is substantially more fragile in aF than in aL.

Formally, we define $\pi ss(aL,aF)$ as the probability of mutual sharing of two randomly matched innovators, dependent on the exclusivity payoffs aL and aF. Figure 6 depicts the empirical observation of $\pi ss(aL,aF)$, which is the expected frequency of mutual knowledge sharing as a function of all 16 (aFxaL)-games. For a given game, the expected frequency results from the probability that L shares multiplied by the probability that F shares.

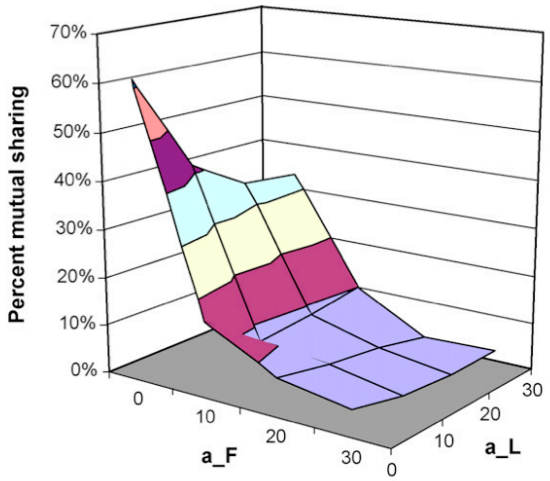


Figure 6: The fragility of knowledge sharing - percentage of mutual sharing

We define the marginal change of the probability πss in the exclusivity payoffs (i.e., $\Delta\pi ss(aL,aF)/\Delta ai$, i=L,F), as the fragility of mutual knowledge sharing induced by i's exclusivity payoff ai, i = L,F. Figure 6 shows that $\pi ss(aL,aF)$ is convex in ai, i = L,F, and more fragile in aF than

in aL. Of particular interest is $\Delta\pi ss(0,0)/\Delta ai$, i=L,F, i.e., the marginal change in the probability of mutual sharing once an exclusivity payoff becomes positive. The expected probability of mutual sharing drops dramatically, once aF > 0. Compare in particular aF = 0 and aF = 10, when aL = 0, which reveals a marginal drop in mutual sharing of 45.9 percentage points. The drop is much smaller in aL than in aF, namely 19.99 percentage points (compare aL = 0 and aL = 10 when aF = 0). Importantly, in the initiation of private-collective innovation, mutual knowledge sharing is substantially more fragile in F's than L's benefits from exclusivity. In other words, F's opportunity costs of sharing represented by exclusive returns threaten mutual knowledge sharing more than L's.

Discussion and conclusion

We investigated the relationship between incentives and knowledge sharing when initiating private-collective innovation. The first part of the study showed that incentives in knowledge sharing give rise to a knowledge-sharing game, a coordination game with multiple equilibria (rather than a public goods game, although mutual knowledge sharing makes knowledge a public good between innovators). Some of these equilibria entail mutual sharing. However, the analysis also revealed that knowledge sharing is fragile: as soon as innovators face opportunity costs of sharing, mutual sharing ceases to be an equilibrium of the knowledge-sharing game. In contrast, mutual concealment is always an equilibrium in the knowledge-sharing game, albeit an inefficient one. The theoretical analysis also revealed that many equilibria entail unilateral knowledge sharing, where one innovator shares his or her knowledge while the other conceals. These equilibria describe genuine conflicts of interest in knowledge sharing, and situations that would prevent private-collective innovation. In the second part of the study, we implemented the model in a controlled laboratory experiment with monetary incentives that took the same form as the incentive structure for knowledge sharing in the initiation of private-collective innovation. The experimental results demonstrated important asymmetries in the fragility of knowledge sharing and, in some situations, much more knowledge sharing than predicted by purely selfish behavior. Taking into account that some people care about fairness, mutual sharing outcomes can correspond to equilibrium strategies even when there is a conflict of interests between innovators.

A few limitations apply to this study. Initial decisions might be affected by repeated interactions with the same individuals. However, the goal of our study was to understand the basic incentive structure of the constituent game that underlies any repeated interaction. For this reason we studied a one-shot game that is not confounded with repeated interactions. Future works should study repeated knowledge sharing games as well as non-repeated games with the possibility of multiple followers. Our model did not include fundamental psychological dispositions explicitly. Context-specific attitudes or processes might influence knowledge-sharing behavior, such as open source ideology, altruism, or OSS development routines. The initiation of private-collective innovation could be facilitated or restrained by cultural factors specific to the context of software development, academia, biotechnology, and so on. However, from a game theoretical point of view one might argue that these factors would be incorporated into the payoffs of the knowledge sharing game (from a game-theoretical view payoffs in games are utilities anyway). To the extent that these added psychological factors do not change the structure of payoffs, our analysis would not change. Moreover, our conventional experimental economics approach of inducing material incentives (Smith 1982) allowed us to observe to what extent social preferences matter on top of material incentives.

This work presents four implications for theory and research. First, to our knowledge, our study is the first to investigate the role of incentives to share knowledge in the laboratory. We provide evidence that economic incentives matter greatly in initial knowledge-sharing decisions. From an economic viewpoint, sharing or concealing knowledge affects costs and benefits for innovators. These results confirm the argument of many authors that it is important to conduct cost-benefit analyses of knowledge-sharing situations (Foss and Mahnke, 2003; see also Takeishi, 2002; Szulanski, 2000). Our study also demonstrates the benefits of an experimental setup for studying knowledge sharing in various situations, in particular where decisions are difficult or impossible to observe in field studies. Building on the results from this study and the findings of other game theoretical work on knowledge sharing (Harhoff et al., 2003; von Hippel, 1987), future research needs to identify the empirical parameters, such as a variety of cost-benefit types, that enable private-collective innovation in various fields. They may include benefits from unilateral sharing such as reputation or signaling.

Second, since many people not only care for their own costs and benefits but also entertain social preferences, knowledge sharing is likely to be affected by inequality aversion, reciprocity, and efficiency considerations. We found that when the leader initiated sharing, the extent of knowledge sharing amongst the participants in the laboratory experiment exceeded the predictions from Proposition 1, and provided empirical evidence for a conjecture in the literature that social preferences impact on knowledge sharing (di Norcia, 2002; Wasko and Faraj, 2000; Faraj and

Sproull, 2000; Orlikwoski, 1992; Kim and Mauborgne, 1998; Kogut and Zander, 1996; Bergquist and Ljungberg 2001). Interestingly, there is also more sharing in situations where inequality aversion and reciprocity predict little sharing, namely when the leader decides to conceal. A concern for efficiency can explain followers' sharing in these situations.

Future studies should distinguish between the various forms of social preferences that impact on knowledge-sharing decisions in the laboratory. This work should also attempt to investigate the transition from a "pure" sharing decision by the leader and the immediate follower, to those situations where interactions between leaders and followers recur, and/or other innovators enter the game. The results from this work will shed more light on the transition from the initiation to the maintenance of private-collective innovation.

Third, our study is important for understanding if and how private-collective innovation can diffuse in OSS and beyond. Recall that the incentive structure in the initiation of private-collective innovation makes knowledge sharing highly fragile. Fragility is likely in any field where an innovator contemplates sharing knowledge as a public good. By showing that the probability of knowledge sharing drops as a consequence of the opportunity costs of sharing, we operationalized and demonstrated the fragility of knowledge sharing. We found that mutual knowledge sharing is susceptible to relatively low incentives for not sharing. This result is important for the initiation of private-collective innovation: if opportunity costs to knowledge sharing are low, innovators are more likely to make their work freely available. In the case of OSS, the findings suggest that opportunities to sell innovations in the software market should influence the decision to publish software under an open source license. However, OSS licenses provide an elegant solution to the problem of the fragility of knowledge sharing in the initiation of private-collective innovation. Such licenses restrict the opportunities to release binary versions of the published code and (unlawfully) license it for a fee to a third party, thus limiting followers' opportunity costs. If innovators share through an OSS license, they know that followers cannot enjoy exclusive financial benefit from selling the software. In fact, innovators may hope that followers use the software and feed back improvements (Raymond, 1998; Lakhani and von Hippel, 2003). Future work should investigate the initiation of private-collective innovation with different license forms. Although across OSS projects in general the most common license is the GNU GPL, as we point out in footnote 2, there are various forms of licenses providing innovators with different weak or strong limitations to appropriate exclusive returns from innovation (for an overview of licenses, see Lerner and Tirole, 2005). Research should investigate the extent to which these licenses create effective incentives for initiating private-collective innovation, too.

One should expect private-collective innovation to flourish only in settings where some forms of license limit followers' opportunity costs ($aF = 0$), and not in others. In the absence of such licenses, one should expect private or collective action models of innovation to dominate. An important emerging area for private-collective innovation incentives is research tools for life sciences and biotechnology—for example, tools for human genome research or genetic engineering of plants. In recent years there has been a dramatic expansion in the patenting of these types of tool by biotechnology firms, and authors warn that unless open source-style licenses are applied to such tools, progress could be halted (Hope, 2008: Jefferson, 2006; Broothaerts et al. 2005). In order to advance their research, universities are often forced to license expensive tools from firms. However the funding for these licenses are in many cases limited or entirely absent. Although universities often contributed to the development of these tools in the first place, researchers are often not allowed to transfer the innovation to public sector institutions or use them to develop products, for example agricultural products for developing countries. Thus, firms are repeatedly accused of blocking innovation rather than using patents to share knowledge on which others can build. Feldmann (2004) discusses how open source-style licenses in biotechnology have the potential to bring innovations in this area into the public domain, and create downstream, non-economic rewards by tapping unused innovation resources (e.g. smaller research labs or individual researchers) that traditional patenting cannot reach. Work exploring open source-style licenses in biotechnology should investigate innovators' incentives to conceal or share knowledge on research tools. An issue here is the extent to which licenses applied to research tools can prevent follower-innovators from capturing and commercially exploiting them. To what extent do open source-style licenses apply to this area, where

innovation is costly, takes longer, patenting is already widespread, and where the opportunity costs (incentives to defect) may be immense from the outset? A thorough investigation of these issues will help researchers understand to what extent the widespread initiation of private-collective innovation can be expected in this area.

The initiation of private-collective innovation may also be of interest to policy scholars. Recently, many countries have adopted a policy of using OSS in governmental agencies and publicly funded institutions. Researchers have investigated the consequences of these policies for public expenditure, firm, and industry competitiveness, etc. (Ghosh, 2005; Casadesus-Masanell and Ghemawat, 2006). Policy research on OSS can be complemented with future work on the initiation of private-collective innovation. An interesting situation would occur if at least one follower-innovator for a software product were a government agency committed to OSS policy. This could potentially lower the opportunity costs of knowledge sharing, implying that innovators would be more likely to make their work freely available in the public domain. Thus, a policy could impact not only on the diffusion but also on the initiation of OSS projects. Our study cannot reach a conclusion about the effects of an OSS policy, but this is an important area for future research.

Fourth, since knowledge sharing plays a crucial role in innovation, the relationship between incentives and knowledge sharing offers fertile ground for the study of the role of incentives beyond the firm-market dichotomy. The firm is usually defined in terms of asset ownership, contracting and incentives (Holmstrom and Milgrom, 1994; Foss, 1996). But there is a continuum of external sourcing methods and hybrids between firms and markets (Leonard-Barton, 1995), which vary in terms of ownership and contracting (OSS communities are one of them). As Holmstrom and Milgrom (1994) comment, little is known about the interdependencies among the defining characteristics of firms. The role of private-collective innovation in software reinforces the need to study non-traditional forms of economic organization (e.g. Hargrave and Van de Ven, 2006). Knowledge sharing and incentives could function as signposts for innovation scholars into organizations that blend the characteristics of firms and markets.

We close with a methodological comment about the use of laboratory experimental methods to study issues in management and organization theory. We argue that studying the causal consequences of incentives in knowledge sharing requires the full observation of all costs and benefits, as well as actual decisions to share or conceal. Since these requirements are impossible to fulfill in the field, our paper was based on a laboratory study where the theoretical model of knowledge sharing was implemented. Experimental results, such as those from our study, complement field investigations. In particular, the tight results from the laboratory can help guide field research, which has the advantage of being more realistic but also the drawback that causal inferences are often unfeasible. Our paper joins recent studies[36] that use laboratory methods to study specific phenomena relevant to management and organization theory that are hard to investigate in the field.

[36] Recent examples comprise the formation of corporate cultures (Weber and Camerer, 2003), bargaining (Zwick and Chen, 1999), deception in organizational decision making (Brandts and Charness, 2003), leadership (Weber et al., 2001), incentives in mergers (Montmarquette et al., 2004) and policies to foster innovation (Meloso et al., 2008).

APPENDIX

Appendix A: Equilibria of the sequential knowledge-sharing game.

The extensive form game of Figure 1 has the following unique normal form representation, where the leader (L) is the row player and the follower (F) the column player. F has to make decisions at two information sets, and has four strategies at her disposal, which we denote as ss, sc, cs, and cc. The Nash equilibria of the extensive form game can be found in its strategic form representation. We assume that players are rational and purely self-interested.

	ss	*sc*	*cs*	*cc*
s	$b_L + v_L - k, b_F + v_F - k$	$b_L + v_L - k, b_F + v_F - k$	$b_L - k, b_F + v_F + a_F$	$b_L - k, b_F + v_F + a_F$
c	$b_L + v_L + a_L, b_F - k$	b_L, b_F	$b_L + v_L + a_L, b_F - k$	b_L, b_F

Table 5: Strategic form representation of the knowledge sharing game

Result A1 (Nash equilibria of the sequential knowledge sharing game if sharing is costly):

If vi > k > 0, then the only Nash equilibrium is the strategy profile (c, cc), i.e., L conceals, and F conceals in both subgames. This holds irrespective of ai, i=L,F.

Result A2 (Nash equilibria of the sequential knowledge sharing game if sharing is costless):

If k = 0, the equilibrium strategies depend on ai, i=L,F. We get the following pure strategy Nash equilibrium profiles (**bold** strategy profiles are subgame perfect):

- *If* $a_L = 0$, $a_F = 0$: **(*s*, *ss*)**, **(*s*, *sc*)**; **(*s*, *cc*)**; **(*c*, *ss*)**; **(*c*, *cs*)**; **(*c*, *cc*)**.
- *If* $a_L > 0$, $a_F = 0$: **(*s*, *sc*)**; **(*s*, *cc*)**; **(*c*, *ss*)**; **(*c*, *cs*)**; **(*c*, *cc*)**.
- *If* $a_L = 0$, $a_F > 0$: **(*s*, *cc*)**; (*c*, *ss*)**; **(*c*, *cs*)**; **(*c*, *cc*)**.
- *If* $a_L > 0$, $a_F > 0$: **(*s*, *cc*)**; (*c*, *ss*)**; **(*c*, *cs*)**; **(*c*, *cc*)**.

Both results follow from the payoffs specified in the strategic form representation.

There is also a host of mixed strategy equilibria. They have the following form. In all mixed strategy equilibria, L plays a pure strategy of either concealing or sharing with probability 1. In other words, in all mixed strategy equilibria it is only F who mix. Their mixing probabilities are as follows:

- If aL=0 and aF>0, then F mixes s and c in both subgames with probability 0.5. L conceals.
- If aL=10 and aF>0, there are two equilibria, in both of which L conceals: (i) in the subgame after L has chosen c, F plays c with probability 0.6 and s with probability 0.4. In the subgame after s F plays c with probability 0.4 and s with probability 0.6. (ii) In the subgame after L has chosen c, F plays c with probability 0.33 and s with probability 0.67. In the subgame after s F plays c with probability 0 and s with probability 1.
- If aL=20 and aF>0, there are two equilibria, in both of which L conceals: (i) in the subgame after L has chosen c, F plays c with probability 0.67 and s with probability 0.33. In the subgame after s F plays c with probability 0.33 and s with probability 0.67. (ii) In the subgame after L has chosen c, F plays c with probability 0.5 and s with probability 0.5. In the subgame after s, F plays c with probability 0 and s with probability 1.
- If aL=30 and aF>0, there are two equilibria, in both of which L conceals: (i) in the subgame after L has chosen c, F plays c with probability 0.7143 and s with probability 0.2857. In the subgame after s, F plays c with probability 0.2857 and s with probability 0.7143. (ii) In the subgame after L has chosen c, F plays c with probability 0.6 and s with probability 0.4. In the subgame after s, F plays c with probability 0 and s with probability 1.
- If aL=0 and aF=0, there are two equilibria, in both of which F mixes between s and c with probability 0.5. In one equilibrium L shares and in the other L conceals with probability 1.
- If aL>0 and aF=0, there are four equilibria, which have the same structure as the equilibria described above for aF>0. In two equilibria, L shares with probability 0 and in two equilibria L shares with probability 1. The mixing probability of F corresponds to those for aL=10, aL=20 and aL=30.

Appendix B: Experimental Instructions

This experiment is about economic decision processes. Please read the following instructions carefully. You are not allowed to talk during the experiment. If you have any questions, please refer directly to the instructor.

The points incurred as income during the experiment are converted to Swiss francs and paid out cash. In the experiment, your income is calculated in points.

One point = 4 Rappen.

Description of the decision situation

- The decision situation in this experiment involves two decision makers.
- The first decision maker decides first: he/she can choose either "left" or "right."
- The second decision maker also faces the choice between left and right. He/she has to choose before he/she knows how the first decision maker has decided. This means that the second decision maker has to choose between left and right, both where the first decision maker chose left and where he/she chose right.
- The relevant decision situation is displayed schematically here, as you will see it later on the screen:

Please enter your decision here:	Decision Maker One: left ○ ○ right			
	Decision Maker Two: left ○ ○ right		Decision Maker Two: left ○ ○ right	
Income Decision Maker One	30	10	x	10
Income Decision Maker Two	30	y	10	10

How decisions are made

The incomes for both decision makers derive from the combination of both decisions. If, for example, the first decision maker chooses left and the second also chooses left, both receive an income of 30 points. If both choose right, they receive an income of 10 points each.

The income for the first decision maker is shown in the first row. The second row indicates the income for the second decision maker.

Altogether, you have to decide for 16 situations. The incomes generated by the decision combinations vary across the situations:

(1) if the first decision maker chooses left and the second chooses right;

(2) if the first decision maker chooses right and the second subsequently chooses left.

In the display above, the incomes from these situations are marked with x and y. Only the incomes x and y vary from period to period across the situations. During each period, the current values of x and y will be labeled in red on the screen. All other incomes remain unchanged across the 16 periods.

How do you make decisions?

- You will be assigned the role of first or second decision maker at random. Your role assignment will be communicated on the screen.
- Through all 16 periods you will be either the first or the second decision maker.
- During every one of the 16 periods of the experiments you will be rematched to another randomly chosen counterpart.
- If you are the first decision maker you have to decide in every period whether you choose left or right.
- If you are the second decision maker you have to decide in every period whether you choose left or right for both possible decisions (left or right) taken by the first decision maker.
- The income from all decision situations will be aggregated and converted to Swiss francs.
- During the 16 periods you will not know how your counterparts decided. You will be informed about your income at the end of the experiment.
- Your income derives in every period from the combination of decisions, given your decision and your counterpart's decision.

• Before the start of the experiment, you will have to answer two control questions on the screen.

Appendix C: Frequency of sharing decisions

In this appendix we document the frequency of individual decisions for each of the 16 games. We also analyze the strategies and equilibria that subjects actually played.

	s_L	s_F after s_L	s_F after c_L
$a_L = 0$, $a_F = 0$	0.84211	0.73214	0.45536
$a_L = 10$, $a_F = 0$	0.61404	0.67857	0.45536
$a_L = 20$, $a_F = 0$	0.48246	0.71429	0.53571
$a_L = 30$, $a_F = 0$	0.44737	0.74107	0.40179
$a_L = 0$, $a_F = 10$	0.55263	0.28571	0.42857
$a_L = 10$, $a_F = 10$	0.34211	0.19643	0.47321
$a_L = 20$, $a_F = 10$	0.30702	0.26786	0.37500
$a_L = 30$, $a_F = 10$	0.35088	0.28571	0.37500
$a_L = 0$, $a_F = 20$	0.45614	0.17857	0.51786
$a_L = 10$, $a_F = 20$	0.34211	0.16964	0.47321
$a_L = 20$, $a_F = 20$	0.25439	0.16964	0.49107
$a_L = 30$, $a_F = 20$	0.18421	0.08929	0.47321
$a_L = 0$, $a_F = 30$	0.42105	0.13393	0.46429
$a_L = 10$, $a_F = 30$	0.23684	0.12500	0.47321
$a_L = 20$, $a_F = 30$	0.21053	0.09821	0.43750
$a_L = 30$, $a_F = 30$	0.17544	0.11607	0.41071

Table 6: Frequency of chosen actions

In section 2 we explained that a theoretical property of the sequential knowledge-sharing game is the multiplicity of equilibria in the absence of out-of-pocket costs of knowledge sharing. The next step is to investigate which of the equilibria are behaviorally relevant. We describe them in Table 7, which summarizes the distribution of choices over the strategy space as a function of the opportunity costs of sharing.

The leader (L) has two strategies, share (s) and conceal (c). The follower (F), however, has four strategies (see Figure 1): he/she can (i) share if L shares or conceals (denoted ss), (ii) conceal if L shares and share if L conceals (denoted cs), (iii) share if L shares and conceal if L conceals (denoted sc), and (iv) conceal irrespective of L's choice (denoted cc). Therefore, the strategy space of the whole game is (s, c)x(ss, sc, cs, cc). Since we have applied the strategy method in our design, we can observe the behavior in the complete strategy space of the 16 games.

Table 7 shows the strategies and—given the subjects' actual choices—lists the expected distribution of strategy combinations for the relevant cases of (aFxaL)-combinations (each row sums to 100 percent). Under the assumption that L and F are randomly and independently matched, the expected distribution is determined by multiplying the observed frequency of s- or c-choices by L and the frequency of F's respective strategy. For instance, in the game with (aL=0, aF=0), L decided for s in 84.21 percent of the cases, and F chose the strategy ss in 38.39 percent. This makes an expected frequency of observing the (s, ss)-strategy combination of 38.39x84.21=32.33 percent. Equilibrium strategies are shaded, and bold letters indicate subgame perfect strategy combinations.

Strategy profiles (c, sc) and (c, cc) induce a mutual concealment outcome ((c, sc) is a non-equilibrium profile, however). When no player has opportunity costs of sharing knowledge, then the expected frequency of mutual concealment is 8.6 percent. When both players have positive exclusivity payoffs, the expected frequency of mutual concealment jumps up to 40.86 percent.

The four strategy profiles—(s, cs), (s, cc), (c, ss), (c, cs)—induce unilateral knowledge sharing, which allows at least one party to benefit if exclusivity is possible. For instance, if (aL>0, aF>0), then we expect unilateral knowledge sharing in 54.63 percent of cases. Particularly interesting is the strategy combination (c, cs), which induces unilateral sharing that results in an unequal payoff benefiting L. In our data, the expected frequency at which F is prepared to resolve indifference in favor of L, if L conceals, is 27.04 percent.

Finally, the strategy profiles (s, ss) and (s, sc) induce a mutual sharing outcome (compare Figure 1). In case no player has a positive exclusivity payoff, that is if aL=0 and aF=0, we observe that the expected frequency of strategy combinations leading to mutual sharing (as an equilibrium) is 61.65 percent. This percentage drops dramatically once at least one player has positive opportunity costs of sharing knowledge. If L alone has positive opportunity costs (aL>0, aF=0), the expected frequency of strategies that induce mutual sharing outcomes is 36.61 percent (16.24 percent are consistent with equilibrium play). If only F has positive opportunity costs (aL=0, aF>0), mutual sharing does not occur in equilibrium. However, we observe mutual sharing in 9.51 percent of all strategy combinations. In case both have positive exclusivity benefits (aL>0, aF>0), the likelihood of strategies supporting mutual sharing drops to 4.5 percent.

In summary, mutual sharing occurs particularly when it is an equilibrium of the knowledge-sharing game. Yet, for the reasons discussed in section 2.4, we observe mutual knowledge sharing even if it is not an equilibrium.

	Outcome is...							
	Mutual concealment		Unilateral knowledge sharing				Mutual sharing	
	(c, sc)	(c, cc)	(s, cs)	(s, cc)	(c, ss)	(c, cs)	(s, ss)	(s, sc)
$a_L = 0, a_F = 0$	5.50	**3.10**	6.01	**16.54**	**6.06**	**1.13**	**32.33**	**29.32**
$a_L > 0, a_F = 0$	15.31	**10.69**	3.53	**11.33**	**19.21**	**3.32**	20.37	**16.24**
$a_L = 0, a_F > 0$	5.45	**22.28**	17.87	**20.28**	4.98	**19.63**	4.54	4.97
$a_L > 0, a_F > 0$	6.98	**33.88**	9.86	12.35	5.38	**27.04**	1.96	2.54

Table 7: Expected frequency of strategy combinations given subjects' choices

Appendix D: An econometric analysis of sharing decisions

An econometric analysis provides further support for Results 1 and 2. Table 8 provides econometric evidence for the impact of the exclusivity benefits aF and aL on (a) F's knowledge sharing decision after L decided to share; (b) F's knowledge-sharing decision after L concealed, and (c) L's knowledge-sharing decision. As the share-conceal decision is binary, we ran a logit regression with the binary variable (1=share, 0=conceal) as the dependent variable.

	(a) Followers' share or conceal decision after leader has shared	(b) Followers' share or conceal decision after leader has concealed	(c) Leaders' share or conceal decision
Dummy $a_L = 10$	−0.0450 (0.0221)*	0.0044 (0.0225)	−0.1701 (0.0263)**
Dummy $a_L = 20$	−0.0218 (0.0219)	−0.0067 (0.0206)	−0.2524 (0.0300)**
Dummy $a_L = 30$	−0.0286 (0.0189)	−0.0596 (0.0216)**	−0.2833 (0.0363)**
Dummy $a_F = 10$	−0.4572 (0.0365)**	−0.0508 (0.0240)*	−0.1914 (0.0311)*
Dummy $a_F = 20$	−0.5685 (0.0353)**	0.0286 (0.0187)	−0.2832 (0.0367)**
Dummy $a_F = 30$	−0.6039 (0.0365)**	−0.0132 (0.0267)	−0.3453 (0.0390)**
Observations	1824	1824	1824
Wald $\chi^2(6)$	183.5**	17.75**	149.73**
Pseudo R^2	0.2113	0.0043	0.0916

Robust standard errors in parentheses
* Significant at 5%.
** Significant at 1%.

Table 8: Marginal effects of logit estimation of the sharing decision

The independent variables are dummies for the respective levels of aF and aL; the omitted benchmarks are aF = 0 and aL. = 0. To account for the fact that a subject's decisions might be correlated across games, we calculate robust standard errors with clustering of decisions at subject level (between subjects decisions are independent by design).[37] Since coefficients of logit estimations are hard to interpret, we report the marginal effects in Table 8, which shows how an increase in ai, i = L,F, influences the probability of sharing.[38]

Column (a) reports the results of how F change their knowledge-sharing decision, relative to the benchmark game where aL = aF = 0. Holding L's exclusivity payoffs constant, we find that the drop in F's knowledge-sharing rates is quite dramatic and highly significant. Relative to the benchmark, the likelihood F will share drops by more than 45 percent, if his/her exclusivity payoff changes from 0 to 10. The likelihood of sharing drops by more than 60 percent, once aF = 30. L's exclusivity payoff does not matter: a $\chi 2$-test cannot reject the null hypothesis that the three dummies are jointly not different from zero (p=0.194). In other words, in their knowledge-sharing decision F do not take L's exclusivity payoff into account when deciding whether to share knowledge or not.

Column (b) shows F's knowledge-sharing decision, after L has concealed. The estimated changes in probability of sharing are small (although in two cases significant) and we do not find a systematic pattern.

Column (c) documents L's sharing rate, relative to the benchmark. L is significantly less likely to share if his/her own exclusivity payoff aL is high, but also if F's exclusivity payoff aF is high.

[37] A random effects panel model yields very similar results.

[38] Specifically, we calculate the marginal effects when all dummies are zero. The marginal effect measures dy/dx for a discrete change of a dummy variable from 0 to 1.

SOCIAL SOFTWARE AND STRATEGY

Stefan Haefliger [a]
Eric Monteiro [b]
Dominique Foray [c]
Georg von Krogh [a]

[a] Chair of Strategic Management and Innovation
Department of Management, Technology, and Economics (MTEC)
ETH Zürich, Switzerland

[b] Department of Computer and Information Sciences, Norwegian University of Science and Technology NTNU, 7491 Trondheim, Norway

[c] EPFL, CDM e CEMI, Odyssea 1.16, Station 5, CH - 1015 Lausanne, Switzerland

Published in Long Range Planning vol. 44 2011 page 297-316

Social software challenges strategic thinking in important ways: empowering creative, independent individuals implies indeterminate and uncertain reactions and creations in support of, or in opposition to, management's original thinking. We build a framework that organizes research on social software, taking perspectives from both inside and outside companies. We use this framework to introduce the contributions to this special issue in terms of strategy, technology, and community and to ask a series of questions for strategy research that pays particular attention to value creation and appropriation, the role of technology both as tool and mediator between managers and users, and the role that management can play in communities, both as leaders and in shaping boundaries.

INTRODUCTION

Aligning interests, motivating contributions to knowledge work, and giving direction to multiple business units and market initiatives represent daily challenges for the strategist in most companies. The diversity of contexts within an organization has led critical management thinkers to suggest that in the presence of multiple initiatives and discourses, the introduction of a new technology has multiple unintended consequences for organization. New technology is regularly subject to power struggles, conflicting goals, and discrepant events (Markus, 1983; Barley, 1986; Orlikowski, 1992; Ciborra, 1996; Leonardi, 2008) which impact on how strategies are shaped within organizations.

Information technology, such as social software, may affect the interaction patterns between organizational members, create new opportunities for knowledge and information sharing (von Krogh, 2002), or unfold the disruptive and possibly change-inducing potential of so called "informational capabilities" (Leonardi, 2007). Informational capabilities refer to an information technology's potential to alter the storage, transmission, and creation of information in an organization. In this respect information and communication technology (ICT) differs from other technologies adopted by organizations. Despite the broad application of ICT and its potential implications for company performance (Powell and DentMicallef, 1997; Tippins and Sohi, 2003; Ho et al., 2011), strategy scholars seldom include the properties of ICT in their theorizing on strategic thinking, firm growth and its boundaries, or the strategies for creating ICT infrastructure (Leidner et al., 2010; Yoo et al., 2010).

The term "social software" grew out of the notion of groupware and computer-supported collaborative work (CSCW) and had been attributed to Clay Shirky, who denoted software that supports group interaction (Allen, 2004; Shirky, 2005). Designed to facilitate individual creativity combined with community building, groupware and social software have led, in their 1.0 incarnation, to novel and significant insights in academic fields ranging from technological innovation, organizational behavior, and management and organization theory, to strategy (Sproull and Kiesler, 1986; Sawhney and Prandelli, 2000; Lee and Cole, 2003; von Krogh and von Hippel, 2006). Today, social software, frequently annotated with Web or Enterprise 2.0, also receives a lot of interest from managers due to its commercial use, increased network functionality, massive mobilization of users in some cases, and growing infrastructure capabilities, such as multi-media streaming online.[39] The more than enthusiastic reception of LinkedIn by investors during their initial public offering in May 2011 suggests that the commercial promise of a business model involving large numbers of users connected through social software inspires investors: secondary market valuations of companies such as Facebook, Groupon, or Twitter are interpreted along similar lines or else discounted as signs of market participants' exuberance.

Social software affects the interaction between employees within and individuals outside the firm, such as members of user communities or customers. In many industries, users of technology, frequently organized in communities, are known to innovate independently of manufacturers (von Hippel, 1988; 2007), and consumers have successfully contributed to innovation and product development organized by firms (Füller et al., 2010; Franke et al., 2010). A perspective that privileges firm-internal matters, which the strategy field has often tended to adopt (Minzberg, 1978), risks overlooking the increasingly powerful and important position that individuals outside the firm hold, particularly when organized in communities (Fredberg, 2009; Dahlander and Wallin, 2006). Here, users and customers set up the governance structures for their communities independently of firms (Markus, 2007; O'Mahony and Ferraro, 2007), often voice criticism toward firms and their products (Kaplan and Haenlein, 2010; Kozinets and Handelman, 2004; Muniz and O'Guinn, 2001), or develop rival products in existing markets (Casadesus-Masanell and Ghemawhat, 2006; Young and Rohm, 1999). Hence, in terms of strategic analysis, users and consumers can be "suppliers," "competitors," or "providers of substitutes," as shown in the examples of widely used Open

[39] For the original definition of Web 2.0 see O'Reilly, 2005. Wirtz et al. (2010: 276) characterize Web 2.0 with four factors: social networking, interaction orientation, personalization/customization, and user-added value.

Source software (OSS) programs like Apache servers or the GNU Linux operating system.

Social software enables the activities of people inside and outside companies actively to shape strategies. The papers in this special issue offer novel, strategic insights on this subject. Social software thus holds an intriguing potential. As consistently demonstrated over the last couple of decades, the actual realization of this potential is anything but straightforward. This special issue unites authors who have gained deep insights into the workings of user communities, their technologies, and the strategic potential that collaborations between firms and communities harbor, in terms of value creation and appropriation. This introductory article contributes to an overall positioning of the papers, drawing on the literatures from fields such as strategy, ICT, technology studies, and innovation. In so doing, we also develop a research agenda on social software that we hope will inspire strategy scholars to continue work in this important area.

Social software creates platforms for self-expression (Schau and Gilly, 2003) and direct interaction between individuals and so facilitates rapid and often spontaneous community building (Culnan et al., 2010). Social software also enables interaction between online consumers and users, their product development efforts (Füller et al., 2010), their mutual rating of ideas and comments (Haefliger et al., 2009), and provides online community members with a basic infrastructure for their work (Lee and Cole, 2003; Ren, Kraut, and Kiesler, 2007; Kohler et al., 2011). Thus, social software becomes an exciting topic for strategy practitioners and scholars. If social software supports management in harnessing the creative output of individuals inside and outside the firm, the deployment and diffusion of such technology may hold considerable business potential. The video game industry, for example, experiences rapid growth thanks to social software platforms (such as Facebook, which has more than 500 million registered users) that serve as alternatives to consoles and enable online gaming involving competition among friends connected through such platforms.[40]

At the same time, the "harnessing" metaphor often employed by companies interacting with communities may be problematic. It might very well suggest too much firm influence. The connotations of control embedded in the metaphor need to be supplemented or even substituted by stronger aspects of cultivation and facilitation. Evidence for successful interaction between firms and user communities is scarce (e.g. Stam, 2009; Stuermer et al., 2009), despite frequently high investments by firms in such collaboration (Dahlander and Wallin, 2006). IBM, for example, invested significant resources into the public development of their Eclipse software development platform for a duration of five years, before development by outside software users outweighed IBM's own development efforts (Spaeth et al., 2010). There are three major implications resulting from the use of social software that favors a broader view of collaboration, extending beyond the company. First, users can assume several strategically important roles for the company beyond their obvious role as consumers of products and services. Most notably, they may supply ideas for product development (see Fuchs and Schreier, 2011), and may offer competing products, as is the case for OSS (see von Hippel, 2007). Second, social software shares with all information technology the capacity to change organizations in unpredictable ways, because it directly alters the way and the location where information is stored, shared, and created (Kling and Scacchi, 1982; Markus, 1983; Leonardi, 2007; 2008). The fact that most "outside members" of social software platforms are unknown to the firm makes it even harder to foresee how ICT will change the organization. Third, users rely on social software to organize within online communities that may or may not be supported by companies, and "develop a life of their own" (Wiertz and de Ruyter, 2007: 390). Understanding which interventions by the company will be perceived as beneficial or obtrusive is key to building lasting relationships with members of such platforms (Jeppesen and Fredriksen, 2006). In the software industry, IBM received credit for their efforts to support the OSS community, whereas Sun Microsystems (now Oracle) was widely criticized for their hesitation to release the source code for Java, the cross-platform programming language

[40] For example, Zynga, the producer of games such as FarmVille, is valued at US$10 billion and is thus significantly higher than, e.g., the game industry incumbent Electronic Arts (US$7.5 billion) according to the Wall Street Journal (February 19, 2011). And while 90% of all users play for free, 10% of the 250 million users reliably pay small amounts for in-game assets and enhancements.

(West and Gallagher, 2006; Vaughn-Nichols, 2009).

These three implications build a framework, which we present in the next section, which organizes research on social software, taking perspectives from both inside and outside companies. We use this framework to locate the contributions to this special issue in terms of strategy, technology, and community. Later, we use the framework to highlight open issues for strategy research building on the contributions by authors in this issue.

TOWARD A FRAMEWORK

Social software has been in use in firms for a number of years but standard or best practice ways of applying social software are not yet visible. Managers and researchers alike still struggle with questions, such as why and how to interpret social software, what are shared perceptions, how to appropriate potential business value, when to enact work practices involving social software, and where to align it with other business processes. For example, the usefulness of social software as an internal communication device is up for debate (see Denyer et al., in this issue). Moreover, the perception of Facebook as a platform to do business is changing the gaming industry. Finally, aligning internal software development efforts with external community development creates new challenges for design science (von Krogh and Haefliger, 2010).

The role of information and communication technology in organizations has been the focus over decades of research (Markus and Robey, 1988; Leonardi and Barley, 2010). Early organizational theorists considered technology to have a unidirectional impact on organizations, forcing management to change some aspect of the organization according to the contingencies inherent in the material features of the technology (Perrow, 1967). This perspective was later challenged by social constructivists who focused instead on the members of the organization who responded to the technology's constraints and to each others' use of the technology (for a review, see Leonardi and Barley, 2010), as well as on the material features of the technology and its role as an actor in organizations (Orlikowski and Scott, 2008; Wagner et al., 2009). This highlights the inherently unintended consequences of technology, which undermine its overly instrumental "deployment" (Rolland and Monteiro 2007). It is this active role of ICT as a mediator between individuals in organizations and between intended and completed actions that complicate strategic deployment and adoption of ICT in firms, even more so when individuals outside and frequently unknown to the company play an important role.

The study of social software in terms of its strategic implications needs to accelerate, since many companies are far advanced with experiments connecting individuals inside and outside the company. At the same time, strategy research should broaden the narrow perspective of authoritative decisions about technology adoption that might miss the influential role technology plays outside the direct control of management. With this special issue and this framework we attempt to follow both suggestions. More specifically, we suggest issues for future research to build on a balanced perspective that takes into account what management can and should influence, combined with an appreciation of consumers' and users' work outside the firm. Two observations about social software may help strategy scholars understand the connection between the perspectives from inside and outside the firm. First, social software shapes the behavior of individuals during evaluation, adoption, early use, and adaptation. Second, and closely connected to the first point, it enables individuals inside and outside the firm to appropriate features of the technology in ways unintended by management or the technology's designers (Markus and Silver, 2008; DeSanctis and Poole, 1994). The case of LEGO (Hienerth et al., in this issue) shows how the adoption of social software enabled a business model where customers co-create new products and commercialize them on the LEGO platform. In the process LEGO had to overcome significant organizational and psychological barriers including the fear of losing control. Stuermer et al.'s (2009) study of Nokia's development of the Open Source Maemo platform reached a similar conclusion.

There are a number of recent contributions in strategy and organization theory that have addressed issues involving social software in the domains of strategy, technology, and community. Table 1 presents the proposed framework and the research published in this special issue. The table distinguishes work that takes a view from inside the company, and studies that focus on similar issues outside the firm. Rather than aiming at an extensive review of this literature, we focus below on distinctive characteristics of the two views that are relevant for understanding the strategic dynamics social software can help generate. First, research that bridges the two perspectives has emerged in the management of innovation (Lamastra, 2009; Fuchs and Schreier, 2010; Capra et al., 2011), but less so in strategy, management, and organization theory (O'Mahony and Bechky, 2008). The articles brought together in this issue bridge the insights that emerge from studying managerial intentions as well as user and consumer behavior inside and outside the firm. Stud-

Table 9: Special issue contributions within a framework for using social software from a perspective within and outside the firm.

	VIEW FROM INSIDE THE FIRM	SPECIAL ISSUE CONTRIBUTORS	VIEW FROM OUTSIDE THE FIRM
Strategy	Implement		Emergence
Value creation	Inviting and empowering customers to contribute to product development (Fuchs & Schreier, 2010)	Burger-Helmchen and Cohendet	Strategic interaction with other users (Kuk, 2006) and learning benefits (Lakhani & von Hippel, 2003)
Value appropriation	Firms' differentiated involvement in communities, dual licensing (Dahlander, 2007), selective revealing (Henkel, 2006), better innovation performance (Stam, 2009)	Hienerth, Keinz, and Lettl	Availability and dissemination of assets under Open Source and Creative Commons licenses (Lerner & Tirole, 2005), or appropriation by user entrepreneurship (Haefliger et al., 2010)
Technology	Deploy		Self-expression
Social software as a tool	Gaining access to creative users, utilizing their judgement and their know-how (Füller et al., 2010)	Denyer, Parry, and Flowers	Use of blogs and community participation for self-expression (Schau & Gilly, 2003) and identity building (Muniz & O'Guinn, 2001, Kozinets, 2002)
Social software as a mediator	Platform-induced biases (Dellarocas and Wood, 2008), groups and user-generated content and behavior as "runtime effect" (Shirky, 2005; Cooke & Bukley, 2008)	Frey, Lüthje, and Haag	Technology architecture signals value (Baldwin & Clark, 2006) and suggests (self-) assignment of tasks and specialization (Yamauchi et al., 2000; von Krogh et al., 2003)
Community	Harness		Belonging
Leadership	Trade-offs between community founding and sponsorship (West & O'Mahony, 2005; Shah, 2006), community leadership costly and complex (Spaeth et al., 2010)	Sutanto, Tan, Battistini, and Phang	Central role of most achieved members of the community (Moon & Sproull, 2001; Raymond, 1999), social skills matter beyond technical savvy (O'Mahony & Ferraro, 2007)
Boundary	Cultural differences as challenge (Pauleen & Yoong, 2001), risk of knowledge leakage (Hustad & Teigland, 2008)	Jarvenpaa and Lang	Firm involvement makes a difference in terms of contribution and motivation (Shah, 2006; Stewart et al., 2006) yet firm recognition matters (Jeppesen and Fredriksen, 2006)

ies limited to a view from outside the firm may bear little business relevance and studies limited to a view from inside the firm may ignore activities by (sometimes unknown) outsiders with significant potential impact on strategy and new business models.

Strategy

Thirteen years ago, Jeffrey Sampler (1998: 349) was already proposing that the availability of critical information for the same market defines industry boundaries, rather than what strategy scholars' considered an industry; firms delivering comparable or similar outputs. Thanks to the free exchange of information over the Internet and the access to social software applications, information that assumes a strategic importance for a market may be available to individuals inside or outside companies. This information may include insights into customer preferences as they are shared in social networks, ideas for new products and services, or information about available substitutes. On the one hand, research on consumer and user communities in the areas of marketing and user innovation focused on a view outside the firm, and showed that consumers and users build communities and organize to achieve specific goals (Muniz and O'Guinn, 2001; Moon and Sproull, 2001; O'Mahony, 2003; Wiertz and Ruyter, 2007). On the other hand, strategy research looked inside the firm to approach the question of how firms can make use of consumer and user communities in the creation and appropriation of value. The results from implementing social software here, however, either tend to be focused on specific positive cases or provide inconclusive results (da Cunha and Orlikowski, 2008, see also Denyer et al., this issue). Moreover, the growing literature on open innovation tends to confine the exchange with external parties, such as suppliers or users, to identifiable and manageable knowledge (Chesbrough, 2003), such as research papers, information about patents or instruments, or to one-directional search in a space of technological opportunities (Laursen and Salter, 2006; Jeppesen and Lakhani, 2010). An exception can be found in new product develop-

ment, where firms have started successfully to empower customers to interact among themselves (Fuchs and Schreier, 2010). Using social software in its simpler (1.0) versions, firms have successfully appropriated value from implementing strategies targeted at collaborating with OSS development communities (Henkel, 2006; Dahlander, 2007) as well as modding communities in the video game industry (Arakji and Lang, 2007). However, so far only Stam (2009) has conclusively linked collaboration with Open Source communities with innovation performance. Frequently, the argument in the literature is that devoting resources to collaboration strategies with user communities would only be beneficial if their contribution paid off for the firm (von Hippel and von Krogh, 2003; Dahlander and Wallin, 2006).

A view from outside the company suggests that social software generates value for individuals because it facilitates interaction with and learning from other consumers and users, helps to build shared identity, and enables joint creation and shaping of technology for own use (Lakhani and von Hippel, 2003; Kuk, 2006, Hertel et al., 2003). Initially, so-called "commons-based peer production" relied on users who connected and exchanged information through a set of rather simple social software tools such as email lists, Internet relay chat, and message forums (Benkler, 2002; Lee and Cole, 2003). While the technology became more sophisticated (2.0), the communities spread and grew: individuals perceived value in collaboration and continued to exchange information in online communities (Ren et al., 2007). Meanwhile, much of the value generated is free and publicly available (including posts in online forums and OSS) and consumers and users take measures to protect this value (O'Mahony, 2003), such as non-profit incorporation, social norms of collaboration, and legal refinement (e.g. licenses under which OSS is made available). Licenses are designed to keep access to information and technology as open as desirable or possible, and they include creative commons and various Free and Open Source software licenses (Lerner and Tirole, 2005, Lang et al., 2009). Furthermore, users turn entrepreneurs by learning from industry experts and recruiting through a network enabled by social software (Haefliger et al., 2010).

The first contribution to this special issue by Burger-Helmchen and Cohendet provides guidelines and examples of how companies in the video game industry foster special relationships with community members outside firm boundaries, in order to gain insights and creative output from collaborating with closely bound and loyal customers. The authors classify communities and their members in order to understand better appropriate firm action to improve the relationships with communities; co-creation of value is a fragile process that depends on motivation and mutual trust. The second contribution by Hienerth, Keinz and Lettl explores the characteristics of user-centered business models, building on well-known cases such as LEGO and IBM. A core contribution lies in identifying successful strategies for integrating users into essential business processes and overcoming internal resistance. The authors elaborate how these processes enable the appropriation of value as part of a new user-centered business model. The paper by Jarvenpaa and Lang compares two distinct platform ownership strategies for supporting content co-creation, using real cases from the music industry. They find that both firm-based and community-based platform ownership designs present viable strategies. The authors suggest that firm-based platform ownership strategies are associated with a moderate level of openness (in terms of content and community membership) and are most effective when importance is granted to the coherence of the co-created content output and opportunities (for community members) to express identity. Community-based platform ownership strategies, on the other hand, are most suitable when novel content and opportunities (for community members) to develop competence are critical.

Technology

ICT is a management tool but, at the same time, it harbors a deeply disruptive potential for organizational change that should not be underestimated. Users inside and outside companies attribute meaning to the functionality offered by a technology that can alter the identity of a technological artifact, such as a search platform or a discussion forum (Faulkner and Runde, 2009), change work practices such as information seeking (Leonardi, 2007), or, as Shirky (2005) puts it, result in a "runtime effect" of ICT. Analogous to software, certain characteristics of the program become apparent as a runtime effect: only after an ICT system is installed and used, can we discern its real impact on the organization. Bridging economics and management, Brynjolfsson and co-authors (2009) discuss evidence of the key

ways in which firms have transformed the firm, supplier relations, and the customer relationships by combining IT with changes in work practices, strategy and products and services. Case studies and econometric work point to organizational complements—such as novel business processes, skills, and organizational structures—as major drivers of the contribution of information technology. The management literature often uses suggestive wording about "harnessing" and "utilizing" users' creative thinking. Management scholars have frequently adopted a perspective on ICT as a tool for gaining access to users' creative output (e.g. Fuller et al., 2010). Less often, authors have focused on the biases and novel forces social software introduces when mediating and organizing work. For example, Dellarocas and Wood (2008) demonstrated the impact of online trading platforms on the interaction between users in a way that resulted in massively overstated user satisfaction. In a study on market research, social software is shown to have the potential to generate insights, due to the links it enables between consumers, whose commenting and rating behavior introduces quality judgments and points to trends (Cooke and Bukley, 2008). In these examples, social software mediates between individuals' activities and collective outcome in ways that are limiting or enabling.

Users perceive social software as a tool for creative expression and identity building online (Schau and Gilly, 2003; Muniz and O'Guinn, 2001). Visibility and peer recognition motivate consumers and users to share their personal experiences with products and companies, and even lead to the development of sub-cultures with specific vocabularies, creative expressions, and behavior (Kozinets, 2002). Conversely, users are shaped by social software, the architecture of digital artifacts, and the specific practices of collaboration that surround and build these artifacts. According to Baldwin and Clark (2006) the architecture of a software can be understood in terms of "option value" where collaboration and contributions to its development are guided by the user's perceived rewards in terms of progress and recognition. For example, users are known to self-select into tasks for a collaborative project in OSS development (Yamauchi et al., 2000) and the specialization of labor in projects follows the logic of an evolving and growing technology implementation (von Krogh et al., 2003). Hence, there are strong links between the architecture of an ICT system and user behavior, including where and to what part of the technology users choose to contribute, how they collaborate and communicate, or even when and where they choose to free-ride on what other users provide.

Contributors to this special issue approach the technology of social software from a strategic perspective, paying particular attention to the motivations and reservations of individuals inside and outside the firm. The article by Denyer, Parry, and Flowers documents the effort to deploy social software within a large telecommunications company. Their story is one of disappointment relative to the high-flying promises of openness and participation. The authors find that the solution implemented did not achieve positive outcomes relative to more traditional methods of communication. The authors offer valuable insights into political processes, such as monitoring and self-promotion, which may have contributed to the dismal reception of the new technology. They show that the problems uncovered do not lie with the technology but with the behavior of users, who need to find a delicate balance of power between the organization's leaders and employees. The contribution by Frey, Lüthje, and Haag studies a search platform for innovative ideas. Social software takes the form of a mediator between individual contributors and firms performing broadcast search on the platform. The authors suggest that deploying such a platform leads to more substantial contributions, when it succeeds in attracting intrinsically motivated individuals with diverse knowledge backgrounds. Both these contributions engage with technology and lead the authors to caution strategists in being too ambitious towards social software and urging them to take users' perspectives and motivations seriously.

Community

Social software is an integral part of the formation of online communities; it enables individuals, who may not be previously linked, to interact and socialize. Part of the strategist's fascination with social software stems from the possibility of accessing a pool of voluntary contributors to strategy, products, services, and business models, who are qualified, motivated, and productive. Realizing this potential requires influence, which is not easy to gain. Dan Frye, Vice President of OSS at IBM, commented on IBM's work with the Eclipse community (quoted in Kerner, 2010): "There is nothing that we can do to control indi-

viduals or communities, and if you try, you make things worse. What you need is influence. It goes back to the most important lesson, which is to give back to the community and develop expertise. You'll find that if your developers are working with a community, that over time they'll develop influence and that influence will allow you to get things done."

The question is, what strategic actions toward facilitating community interaction are possible, for whom, and at what stage (Thompson 2005)? Leadership in an online community is fragile because gaining influence takes years of commitment and investment (Spaeth et al., 2010) and the involvement of companies may change community members' motivation (Shah, 2006; Stewart et al., 2006). For example, companies must decide whether to found a community or to sponsor an existing community (West and O'Mahony, 2005). Both options involve trade-offs with regard to control, influence, and costs. Users may look beyond the deployment of social software to consider joining existing social networks that span across and beyond companies. Thus, it becomes important to understand the governance structures of online communities in order to appreciate the differences and potential risks when applying leadership practices established in a corporate context (Markus, 2007; O'Mahony and Ferraro, 2007). Leadership in user communities is thought to emerge from a meritocracy where technical achievement and boundary spanning is rewarded with power (Fleming and Waguespack, 2007). O'Mahony and Ferraro (2007) drew a refined picture by showing that technical skill alone does not lead to powerful positions: social skills of mediation and negotiation among community members predict future leaders more reliably (see also Fleming and Waguespack, 2007; Collier et al., 2010).

Community boundaries form around individuals, frequently volunteers, who "interact over time around a shared purpose, interest, or need" (Ren et al., 2007: 378). Beyond a general understanding of the risks in social software, such as "knowledge leakage" (e.g. Hustad and Teigland, 2008), the current strategy literature offers little guidance for firms on how to manage community boundaries. Central questions involve the selection, joining, and adherence to norms in existing communities and how this relates to staffing, task allocation, or business process involvement. Professional communities play an important role in the early stages of alliance formation (Rosenkopf et al., 2001) but comparable research on online communities is largely lacking. A potential recourse for strategy scholars may be the literature on virtual teams, which emphasizes the role of facilitators and cautions about the risk inherent in spanning different cultural contexts (e.g. Pauleen and Yoong, 2001; Martins et al., 2004). This risk may grow when linking corporate and non-corporate contexts.

The practice of setting up mechanisms to protect intellectual property reveals that users, like companies, are concerned about losing control over their work (O'Mahony, 2003). However, users often operate in a context of private-collective innovation outside a corporate hierarchy and without labor contracts that regulate their contributions to the community or company (von Hippel and von Krogh, 2003). Because of this, researchers have devoted comparatively more attention to the motivation of users with respect to community boundaries than to leadership issues involving the firm. The motivation to contribute to the community seems to be affected by whether or not companies are involved and sponsor the community (Shah, 2006; Stewart et al., 2006), and the extent to which the firm explicitly credits and recognizes users' contributions (Jeppesen and Fredriksen, 2006).

Two contributions in this special issue deal explicitly with the interactions between firm and community. Sutanto, Tan, Battistini, and Phang test a model of emergent leadership in a setting where users interact and develop network ties. The model predicts perceived leadership from the interaction patterns of users and may provide strategists with specific insights and potential levers about companies interacting with user communities. The work by Jarvenpaa and Lang focuses on boundary management in platform-based online communities by taking a holistic perspective on platform owners and users who form communities. Based on findings from a case study that compares a firm-sponsored fan community and an autonomous community of practice in the domain of digital music remix, they discuss the interactions and interdependencies among organizational boundaries and tradeoffs between openness and control. They argue for integrated management of power, identity, competence, and efficiency boundaries as a necessary condition for achieving community goals and managing what the authors call the "generative capacity" of online communities—that is, their

ability to create and innovate content around the members' shared purpose.

This special issue on social software assembles works that span perspectives inside and outside the firm by studying topics of relevance to strategy and placing the emphasis on the role of consumers and users. Next, we build on these contributions by formulating an agenda for future research.

	OPEN ISSUES FOR STRATEGY RESEARCH	SPECIAL ISSUE CONTRIBUTORS
Strategy	Co-create	
Value creation	What are organizational conditions for successful value co-creation? What are the conditions for mutual buy-in of internal and external stakeholders? How can long-term relationships between firms and users emerge and sustain?	Burger-Helmchen and Cohendet
Value appropriation	Are there generic strategies for firms to appropriate value from co-created assets? What are successful business models for entrepreneurial firms engaging in co-creation of value with users? How could information flows, altered by social software, open new opportunities for value appropriation?	Hienerth, Keinz, and Lettl
Technology	Contextualize	
Social software as a tool	Are there optimal use patterns of social software in certain industries or for specific business processes? How can social software be used to catalyze organizational change? When is sourcing and when in-house development and adaptation optimal?	Denyer, Parry, and Flowers
Social software as a mediator	How can power relationships during implementation be made transparent? How can the organizational impact during the implementation phases be negotiated to proceed fairly? How can management best assimilate, respect, and act on ethical concerns of users?	Frey, Lüthje, and Haag
Community	Co-exist	
Leadership	Can governance structures of user communities be emulated by firms? Can economic arguments buy influence and what roles do other values play? When is firm leadership optimal and when to defer to community leadership?	Sutanto, Tan, Battistini, and Phang
Boundary	What are effective policies and strategies for access to communities and social networks? Which areas of business should best be involved with managing the changing community boundaries? Can different community platform ownership structures be designed and network positions be build and, if yes, which ones are desirable for whom?	Jarvenpaa and Lang

Table 10: Open research issues for strategy research and social software

OPEN ISSUES FOR STRATEGY RESEARCH

There are several open issues regarding social software that deserve the attention of strategy scholars. Many of them start with a practical appreciation of the business implications of this technology: practical technologies for recruiters may help human resource management refine their frameworks for talent management and succession planning; new ways of storing, accessing, and locating patient data may bring about not only personalized medicine but also changes in health management systems; best practices of compensating resourceful users boost new product development initiatives.

Generally, strategic management is concerned with issues like firm survival, the allocation of resources across business units, or the creation of novel business models. In what areas other than those treated in this special issue does social software impact on these questions, and how? To begin exploring this question, it is worth considering the larger ramifications of social software. While refined definitions of social software may moderate or limit disruptive effects to specific business processes, the logic builds on what Leonardi (2007) termed informational capabilities of information technology. Organizations need to grapple with fundamentally indeterminate effects when introducing social software at many levels. The idea of a "runtime effect" of social software (Shirky, 2005) is analagous to the role the system environment plays in the execution of a software program. Adoption, use, and adaptation of a new technology, such as social software, provide contexts in which organizational actors define what a technology means and what it can do for them before and during action (see Leonardi and Barley, 2010 for a review). Hence, the "management" of social software becomes an ongoing task that incorporates the user's role and adapts strategy according to negotiation and structuring of work. Against this background, we develop a series of questions for strategy research that pays particular attention to value creation and appropriation, the role of technology both as tool and mediator between managers and users, and the role that management can play in communities, both as leaders and in shaping boundaries. Table 2 contains the questions evoked by the framework.

Co-creation and appropriation of value

The creation of economic value involving consumers and users connected through social software may depend on organizational structures that support their work. These individuals may or may not be members of the same organization. Yet, the new links between individuals, the exchange of information, and the potential to adhere to the norms of such a network may generate opportunities for knowledge sharing and joint work inside an organization that could be very valuable, or even disruptive to existing ways of creating value. The first question regarding organizational structure touches upon fundamental issues in strategy: which parts of the hierarchy remain intact and which may change? How might social software impact on formal and informal organizations and their interaction? How are decision rights allocated among members in business processes with open networks and free flows of information? Who has the authority to interact with external users? What are the "hidden costs" of changes in organization structure?

The issue of value creation has an important time component, in that co-creation between firms and outside consumers and users involves building trust, providing mutual support, and bearing joint questioning. If social software is to grant access to members from outside the company, the meaning of a "common purpose" may change. Some consumers show extraordinary loyalty to brands and products over a long period of time. Creating a shared purpose relating to a brand or a product could be a productive way of activating value co-creation. This may be costly and time-consuming—and the question of what supports mutual buy-in remains open to research. In their seminal study, Jeppesen and Fredriksen (2006) showed that explicit recognition of outside contributions had a positive impact on value creation.

Value appropriation requires relatively exclusive access to an asset or complementary assets from which products or services can be derived. The growth of business models that contain some "free" elements, and the use of advertising to collect revenue (McGrath, 2010), indicate that appropriating value from user-generated content may become easier. However, the creative commons family of licenses may lead to the growth of the number of domains where appropriation of others' work becomes less straightforward, and companies can no longer count on unsuspecting users who, sometimes ignorantly, pass on the

rights to their intellectual property (von Hippel, 1988). With the growing awareness of intellectual property infringements, we also expect more public awareness of ownership. The creative commons movement actively educates users about their rights and advocates their making a conscious choice about how to license creative work.[41] Based on their study of music remix communities that have adopted creative commons content licenses, Jarvenpaa and Lang (in this issue) argue that sustainable community-based participation in deeper forms of co-creation practices—that is, creating multi-generational product derivatives—requires new, nonlinear kinds of authorship certification. In particular, they point to the incorporation of software-automated attribution trees into the creative commons license design as a possible solution to recognizing complex authorship structures and providing an incentive for community members to continue to contribute content to co-creation communities (in the example of cultural goods). In software, OSS licenses limit the possibility of users and firms appropriating the rights to software components for re-sale. Important works by Henkel (2006) and Dahlander and Magnusson (2007; 2008) have classified a series of strategic approaches to this difficulty encountered by software companies.

Future research may uncover generic patterns in business models that take advantage of assets co-created with consumers and users. The work by Hienerth and colleagues in this issue takes a major step in that direction. Following McGrath (2010), who suggested that successful business model innovations are discovery driven, Hienerth and colleagues suggest that the issue of "runtime effects" of ICT may even prove to be an advantage for firms that experiment with technology like social software. Once deployed and subject to adaptation, social software platforms may evolve in unpredictable directions. McGrath (2010: 254) points out that business model experimentation takes place across and within companies. In this way, the use of platforms such as Facebook may alter information flows across and within firms, leading to new opportunities for products and services. Consider Zynga, which produces online games: friends already connected via a social software platform (Facebook) may compete free of charge against each other in online games or acquire certain in-game assets for improved performance, and so on. Cross-promotion activities among games published by Zynga may retain customers or introduce further products and services as the user base grows. The notion of business model portfolios (Sabatier et al., 2010) could be a promising starting point for scholars who want to theorize about complementary strategies for value appropriation using social software.

[41] For further information about creative commons see http://creativecommons.org/about/.

Contextualizing social software

From the perspective of changing opportunity structures and informational capabilities, social software is both a tool and mediator in organizational processes. Maintaining balance and achieving specific goals from a managerial point of view entails paying attention to four factors: context, power, ethics, and trust. All these factors deserve more explicit attention in future research to support strategists. First, social software is applied to a specific organizational context or business process. There is a choice of maintaining and supporting an existing context or accommodating work involving social software. Are certain business processes more amenable than others to work practices involving social software? What part do contingencies such as hierarchical information barriers, openness to new organization members, privacy issues, or prior communication patterns play? Does adaptation of the social software change the way it's received in the organization? Does one specific type of work fit better with social software than others? Can strategy processes be opened to outside participants through social software? Strategists should remember that ICT can be heavily customized or designed in-house. Hypothesizing about contingencies and adaptation after adopting technology may pave the way toward creating a favorable context for using social software in a way that can be perceived as successful by both users and management. Case studies of more or less successful implementations of social software along the lines of the contributions in this special issue may help to identify additional context factors.

Second, power struggles are an important element in a number of areas of technology management, from the viewpoint of institutional theory (Hargrave and Van de Ven, 2006) as well as from an organizational perspective (Leonardi and Barley, 2010). Proponents of specific technologies form networks (Garud, 2002) or change institutions (Hargrave and Van de Ven, 2006) by lever-

aging and applying legitimacy and framing strategies to supersede their opponents. On a micro scale, actors within one organization or community may dominate others in defining modes of use and work practices involving social software. Leonardi and Barley (2010) suggest that because the construction of meaning and organizational change occurs at multiple levels and phases of ICT implementation, and because its outcome is indeterminate, both human activity and the material features of the technology are significant for the outcome of organizational change. Power struggles may well determine the result of strategic initiatives and challenge strategic management in terms of organizational justice and fairness. A pertinent question in this regard is who is allowed to access information on social software platforms, and for what purpose (for a review see Colquitt et al., 2001).

Third, social software can be perceived as a mediator between groups of users and their respective positions. As a platform for exchange, a filter of information and knowledge, and as facilitator of organizational change, technology inevitably bears values and sides with certain perspectives that may reflect the organization only partially or privilege certain (powerful) individuals. Consumers can become fiercely critical of companies, management, or other employees (Kozinets and Handelman, 2004) and voice criticism even while generally advocating the brand they criticize (Muniz and O'Guinn, 2001). Social software may suddenly create opposing factions that were previously hardly aware of each other. ICT may act as platform for the voices of consumers, users, or developers who loudly and explicitly vent what they could not say before or what went unheard by management. Apart from information flows and employee motivation, such confrontation may require ethical deliberation from the strategist before and during the implementation of social software. Where does learning end in user communities and where does disruption for the company begin? What is the correct and appropriate level of respect toward emerging criticism, internal and external? How and when can social software be integrated into the work practice and become a balanced platform, guarantee equal access, or prevent uneven coverage of organizational events? The process by which social software co-evolves with organizations is strategically important, and the opportunities and limitations in managing and mediating co-evolution deserve more attention in future research.

Fourth, the development of social software as a new step in building virtual relations has given a new edge to the issue of trust. It goes without saying that fraudulent behavior, forgery and pretence have not suddenly been spawned by the virtual world and social software. Questions about what is the original and what the copy, not to mention the evaluation of informational goods that are the object of commercial transactions, have given rise to trust issues and highlighted how crucial trust-building mechanisms are to the functioning of markets and communities since the beginning of time. But the development of virtual relations and social software has increased the need for new trust-building mechanisms. What is at stake here is the entire range of mechanisms that facilitate interpersonal and interorganizational transactions, given the new conditions for knowledge transactions and exchanges: increasing specialization, increasingly asymmetrical distribution of information and assessment capabilities, ever greater anonymity among interlocutors and ever more opportunities for identity theft. Clearly, new methods need to be devised to "certify" the knowledge circulating through virtual relations within a context where inputs are no longer subject to control.

Contextualizing social software means studying it as both the tool and mediator of organizational change that is triggered, facilitated, and aided by management. "Contextualizing" implies a process of construction, where users form networks, communicate across boundaries and exchange information that may alter their identities and work or question power relationships. That is why contextualizing social software may make power relationships transparent and bring forth ethical issues that researchers can analyze in the nascent structures of organizations. Actionable strategy research gives managers insights into other organizations' experiences of the demands put on them by very sophisticated or recalcitrant users, internal or external to the organization. On the one hand, internal users may undercut hierarchies by way of informal communications via social networks; on the other, managers scan Facebook entries before hiring. Issues pertaining to power relations and ethics run in several directions and call for research that makes these issues transparent and relates them to technology. Such research should also balance the perspectives between management and users, or internal and external company stakeholders.

Co-existing with communities

Social software plays an instrumental role in facilitating group work and bringing individuals together to form communities. Individuals gather around a shared purpose or attach to other members (Ren et al., 2007) resulting in communities that produce new technology (Sawhney and Prandelli, 2000) or celebrate certain forms of consumption (Muniz and O'Guinn, 2001). Online communities are organizations in their own right that incorporate and govern their work (O'Mahony and Ferraro, 2007), enable joining and specialization of labor (von Krogh et al., 2003), and allow firms to sponsor or regulate work (Shah, 2006; Bonaccorsi et al., 2006; West and O'Mahony, 2005; Capra et al., 2010). The value of social software-enabled communities for business seems obvious in terms of the knowledge they develop and conserve (Brown and Duguid, 2001). There are many types of online community working with various purposes and breeding all sorts of interests and passions. Can companies emulate the best of the governance structures of online communities? And if so, what type of community should serve as a template for learning? Or, more radically, will leadership need to be fundamentally recast in terms of open-ended notions of governance (Hess and Ostrom 2007)? The relationship between firms and online communities is not well understood in organization theory, where outlines became visible for such a relationship to communities located within the perimeters of the firm and operating face-to-face, yet independent and "bottom-up" (Thompson, 2005). The attempt to gain influence in a community may be a struggle; and exerting influence may bring uncertain outcomes for user motivation and user identification with the community's purpose. The study of leadership that bridges and connects firms and communities is an open field for management research.

Similarly, the discussion of community boundaries raises questions about the purpose of a community, its membership base, and its dynamics. First, purpose may play a central role within the work practice and life context of the individuals who become members of the community (Muniz and O'Guinn, 2001). Communities bear and develop crucial knowledge in organizations (Brown and Duguid, 2001) and there is no reason to believe that communities whose users span organizational boundaries fall short on their ability to develop, protect, and share knowledge (e.g. Lerner and Tirole, 2002; Sawhney and Prandelli, 2000). Purpose and membership seem tightly linked, not only for the ex post definition of what the community is about, but because individuals working on similar issues perceive the need to exchange and learn from their peers and mentors (Lave and Wenger, 1991). While this observation is general and pervades studies of collective action (Oliver, 1993; Ostrom, 1998), it translates into a series of questions regarding the use of social software whose technical implementation is more or less cost-free via the Internet. If firms provide opportunities to interact, individuals sharing similar interests are likely to pick up and exchange information. Importantly, the possible effects of a community on work practices within an organization are a direct consequence of the relatedness and bond that move an individual to join a community in the first place.

Consider the fictitious example of a social software platform (such as LinkedIn or Xing) for the recruitment of management talent within a specific industry. Naturally, prospective talent will rush to become visible on the platform, and so will recruiters. Given an open political and cultural context it becomes easy to see how labor market participants within that industry may join such an emerging community to discuss the firms' strategies, voice their ideas, and challenge each others' ideas. The platform may represent both a labor market opportunity as well as a branding and reputation challenge for participating and sponsoring firms. What policies should accompany the implementation of social software for such a platform? The issues include the eligibility of community membership and the authority to set boundaries, both internal and external. Co-existence with user communities means that authority for such policies is either shared with, or delegated to, members outside the organization, particularly if the company is only a marginal member.

Last, the research issues become even more pronounced when we consider the dynamic properties of community boundaries. There is particular value in search that bridges knowledge domains (Poetz and Prügl, 2010; Laursen and Salter, 2006) and, thus, in community membership that expands in unpredicted ways. On the downside, reputation risks increase because of the nature of social software applications, as we discussed above (Barwise and Meehan, 2010). Research on the dynamic properties of community boundaries is needed, particularly regarding communities that extend beyond the boundaries of one com-

pany. Jarvenpaa and Lang (in this issue) suggest that creating both enabling and constraining boundaries is essential for the sustainability of the communities observed and that these (interdependent) boundaries need to be managed holistically and renegotiated with the community on an ongoing basis. This raises the important question of the link between firm sponsorship and community boundaries, which has not been fully explored. The presence of a company in a user community may not only affect members' motivation and work practices (Shah, 2006), but also the membership dynamics and growth of the community.

Conclusion

Social software challenges strategic thinking in important ways: the articles in this special issue show strategy practitioners meaningful ways to deploy social software successfully, and strategy researchers which critical challenges deserve more attention. In our introduction we have summarized the open research issues along three dimensions that we believe are critically affected by the massive changes to everyday work in organizations, due to growing use and acceptance of social software within and across companies.

First, value creation and value appropriation can gain momentum through interaction with consumers and users inside and outside the firm. Referring to collaboration with users, Jarvenpaa and Majchrzak (2010) talk of vigilant interaction that involves simultaneous knowledge protection and sharing. The balance between sharing knowledge with consumers and users and protecting knowledge assets to appropriate value is subject to experimentation in practice and ongoing research in strategic management and organization theory. Users who had been careless about the rights to their content and innovations (von Hippel, 1988) are becoming more aware of available licenses for their intellectual property (ease of use of creative commons and Open Source licenses) and less sensitive about their privacy (using Twitter and location-based services such as foursquare or LocalUncle). A logic of co-creating strategy may extend the notion of emergent strategy to very active and loyal customers and users outside the company, and create opportunities for strategists who understand and internalize the two perspectives of inside and outside the company.

Second, as a technology, social software challenges not only competitive dynamics in industries but also the structure of organizations. With the increasing digitization of products and services, interaction among consumers and users becomes easier and cheaper. Our behavior as purchasers of e-books depends on the device we use for reading them (mobile or at home, etc.) and recommendations from friends and strangers. The competitive landscape is shaped by what Yoo, Henfridsson and Lyytinen (2010) call the layered modular architecture, because the choice of a platform—staying with the e-book example, is a choice of both hardware (e-book reader) and social network (recommendations). Thus, social software may appear to be a tool of strategic choice and at the same time a mediator of relationships between the firm and users inside and outside the firm. Frey, Lüthje, and Haag (in this issue) make the point that empowering and restricting the user goes hand-in-hand with receiving substantive user contributions. This balance is also a question of power relations; the design and implementation of social software needs to take into account that these relations can substantially alter or disrupt organizational processes (Leonardi, 2007).

Third, communities grow and build on social software applications that enable users and consumers to interact. Two choices impact strategic thinking: leadership and boundaries. To what extent should management lead a community and to what extent should strategists influence the extent of growth and influence of the community? Again, both decisions are constrained and enacted as part of a balancing act that makes the community possible in the first place. A logic of co-existence can guide strategic thinking when deciding about sponsoring a community by setting up social software infrastructures and sharing knowledge. The contribution of Denyer and colleagues in this issue alerts management to the pitfalls this can entail. The same logic may guide leadership that can be shared or distributed across community members who stand out independently of, and possibly outside, the firm.

A strategic approach to social software should start with the insight that empowering creative, independent individuals implies indeterminate and uncertain reactions and creations in support of, or in opposition to, management's original thinking. New business opportunities abound and an experimental approach to strategy (McGrath, 2010) may be guided by the signposts erected by successful companies, who maintain long-term relationships with their users. A number of these are described and analyzed in this special issue.

3. Technology

> *"The primary driver of that [reuse] decision is making the project the best it could be. The fact is that we're mortal and we don't have an infinite amount of time to rewrite everything. So even if the other project's code isn't perfect but good enough, you're simply going to use it because if you've got a thousand bugs to fix you don't want to spend the next year rewriting all the software you could just use from someone else."*
>
> *Interview with Mark Gilbert of Abiword, by phone, March 24, 2004*

Technology is fundamental to collaborative innovation. Historical examples may recount cases of collaborative innovation without reliance on ICT (Allen, 1983) but it is the advent of the Internet, IRC, email, and, particularly, groupware and computer supported collaborative work (Allen, 2004; Shirky, 2005) that is of interest for organization and management science today in order to understand how collaborative innovation unfolds and is organized by those involved in its unprecedented speed. The second role that technology plays is the focus of innovation. For reasons of economic relevance and the accessibility of data and intermediary knowledge artifacts, as explained below, this thesis favors technological innovation without completely ignoring other innovations, however. This section contains three essays that focus on two levels of collaborative innovation from the perspective of technology: first on the collaborative groundwork by asking what supports knowledge reuse in the form of code reuse in open source software development, and, second, on the design and integration of strategic information systems.

The logic that guided the study of technology in collaborative innovation started with a series of close analyses of how developers work when collaborating over a distance and over the Internet, using open source software development as the empirical context. We asked: how does collaborative innovation differ from innovation within one organization as described in the textbook examples (for software, see for example Cusumano, 1991)? And how do the structural features of an evolving technology influence the allocation of work effort in collaborative innovation (von Krogh, Stuermer, Geipel, Spaeth, Haefliger, Baldwin, 2011)? The first essay in this section documents our findings related to knowledge reuse in open source software development.

Moving from the development of specific technologies to the design of systems, the second essay in this section takes a look at the interaction between designers of information systems inside firms and developers of systems and system components outside the firm in order to understand the design process, as a special case of innovation, when it becomes collaborative.

The third essay is empirical again and looks at the connections between projects that each build components of a larger system that is being integrated without anybody assuming the role of systems integrator. A close analysis of collaborations across projects reveals that the code is part of the social fabric that developers use to connect with each other and to connect their own work with others' work in the system. Developers build interfaces to bridge components following a negotiation process with users who specify the technical requirements of the interface (and the system) by reusing existing code and forking code to enable experimental, organizational as well as technical, branches.

There are important, current examples of collaborative innovation that focus on low-tech innovations such as Wikipedia (Garud et al., 2007) or cumulative problem solving and idea generation (Füller, Matzler and Hoppe, 2008; Jeppesen and Lakhani, 2011). This thesis privileges the study of technological innovation for a few reasons, none of them being an exclusive justification for studying technology, rather in combination the study of technological innovation holds specific promises for understanding the organization of collaborative innovation in close touch with the details and intricacies that collaboration processes entail. First, technology as a source of innovation drives market success in many industries including software (Brynjolfsson and Hitt, 2000; Clark and Fujimoto, 1991) and entertainment

(Fisher, 2004; The Economist, 2011). The development of new technology is economically and socially relevant.

Second, collaborative innovation is frequently cumulative when focusing on technology development, an extreme case being the GNU Linux operating system that has been in continuous development since 1991. Technological artifacts, such as software code, carry explicit knowledge and can function independently as modules, components, or preliminary systems. Depending on the openness of the system, see below, the cumulative nature of software means that certain sections of the software "stack" can be reused and run in or as part of another system for a user who may or may not have contributed to its development. A similar argument can be made for text (Wikipedia), design blueprints (fashion, sports, or the example of Open Source Ecology used as illustration above to design heavy machinery) or entertainment (production procedures in Machinima, for example). Observing the cumulative and reusable characteristics of technological artifacts allows for a certain granularity in the study of collaborative innovation because the organization of the innovation process becomes more visible and traceable through he technology that is being assembled or created. Adding, reusing, fixing, or changing a specific software code file leaves a mark on the overall system, permits others to evaluate and react on the changes made and triggers further tests and use cycles. It is not surprising that technology development occurs in cycles of evaluation and implementation but that the cumulative nature of the activity, when carried out in a collective, allows an observation of the organization of the activity.

Third, technology, such as software code, can be made publicly accessible and traceable through its development stages thanks to licensing it as open source. The same may be true for artwork and text under creative commons. However, the history of Free and Open Source licenses has paved the way for artwork and text to be published with the freedom to use and share and redistribute the work, for example via the Creative Commons licenses[42]. The important point, here, is the access and permission to use. As legal scholars have pointed out, collaboration on a broader, societal scale will bring welfare to society through innovation if, and only if, prior work is accessible as freely as possible and if the activity of reuse is not sanctioned or discouraged (Lessig, 2004; Lee, 2008). With access and the permission to use publicly available technology new users can join and become collaborators (von Krogh et al., 2003). The details about the organization of the innovation process can be studied when the source and destination of the technological artifact can be observed: a situation ideally available in the case of open source software development.

Fourth, the development of technology is traditionally the domain of research and development departments in companies or applied science educational institutions. The advent of Free and Open Source software development by volunteers created a puzzle for organization scholars (Lerner and Tirole, 2002) who struggled to explain why and how volunteers collaborated on a long term basis to create complex technology without being co-located or knowing each other from face-to-face interaction. Thus, while other forms of collaborative innovation matter, it is technological innovation that posed the salient questions to organization scholars regarding motivation, coordination, and competitive dynamics (von Krogh and von Hippel, 2003).

This thesis makes only a small contribution to an agenda that focuses on technology when studying collaborative innovation. The difficulties and challenges that developers face when collaborating over a distance and when facing complex tasks are considerable and, for researchers, spawn a long agenda for organization and management science. The growing practice of individuals to reuse works that are free as well as protected by intellectual property rights (Lee, 2008) shows the need for better understanding how collaboration occurs with and through technology. The next section on social practice will pick up the interplay of technological and social process in more detail. However, starting from a perspective of technology development, there is an urgent need to better understand what enables individuals to select their point of departure in collaborative innovation and how this decision interacts with the architecture of the technology under development (Baldwin and Clark, 2006). From the perspective of the firm, how can non-contractible resources, such as voluntary contributions, extra effort, loyalty, creativity, identification, advocacy, or sacrifice be attracted and retained?

[42] For more information: http://creativecommons.org/

A structured approach to thinking about the difficulties involved in initiating and coordinating collaborative innovation may start with a simple yet frequently applied typology of "open systems"[43]. They are markets for technologies, open standards, and open source. First, markets for technologies enable transactions involving clearly defined knowledge goods such as patents and licenses (Arora et al., 2001). The notion of open innovation extended this idea by making the use of technology markets a strategic priority for companies when developing innovations (Chesbrough, 2003). Open innovation does not explicitly exclude openness for ideas under development but makes it very clear that company boundaries can be crossed only if sufficient information is available for contracting (Chesbrough, 2003). Contracts may go either way, the company can outlicense knowledge and receive royalties from users of the knowledge or the company may inlicense knowledge in return for a fee or compensation to the creator or broker. Hence, what crosses company boundaries in the funnel leading from idea to marketable product is contractible knowledge. However, this may only be a perceived limitation as recent efforts by companies in the field of mass customization set up contracting schemes that define the ownership of ideas to be created at the outset (Ogawa and Piller, 2006; Füller et al., 2007). The term open innovation tends to be used more broadly to cover also initiatives by companies to set up platforms to interact with consumers and users (Franke and Piller, 2004; Fuchs and Schreier, 2011), a case categorized under open standards discussed next.

Second, the term open standards can be used as shorthand for systems designed to enable interaction on an interface that accepts and gives back knowledge, from product ideas to software applications to entertainment. Admittedly a large category of systems, they cover a host of activities aimed at generating visibility in user communities, diffusing the reach of proprietary technologies, or sourcing talent and ideas through innovation contests. The systems have in common that they rely on a standard or interface that accepts knowledge of a certain format without pre-defining its content. Before going further, this claim needs some clarification.

Open standards, and standards in general, can create platforms that function as ecologies for multiple businesses (Gawer and Cusumano, 2008). The term "open standard" can refer to the requirement of a reference implementation in open source software development or to the transparency of the standard setting process, as used for example by the World Wide Web Consortium (W3C). Business ecosystems built around a central standard or platform include the Apple App Store, Facebook, Microsoft, the Open Handset Alliance, and many others in industries ranging from telecommunications, to automobiles and electronic payment systems (Gawer and Cusumano, 2008). Social networks, such as Facebook, today substitute for consoles in the video gaming industry, which goes to show that sufficiently large platforms can fundamentally change the way industries work. Android, the operating system for mobile devices produced by the Open Handset Alliance (Google, HTC, Qualcomm and others) rivals Apple as a standard with more openness in terms of access to software source code. This example points to the open question of whether more open standards (Android) may prevail over less open standards (Apple) and we are witnessing a struggle between standards on a global scale.

Network effects play an important role in the competition between platforms (Economides and Katsamakas, 2006, Eisenmann, Parker and van Alstyne, 2011) but for innovation learning and access to knowledge are needed as well, that is, access to fluid, partly non-contractible resources fundamental to innovation. For the case of online communities, Faraj, Jarvenpaa, and Majchrzak (2011) argue that these resources are passion, time, identity, disembodiment of ideas, socially ambiguous identities, and temporary convergence. Open standards enable collectives of firms and individuals to set up rules for collaborating and structures (such as a technological architecture) to locate points of relative interest: the architecture of the platform can inform contributors about the value of their contributions before actually implementing them (Baldwin and Clark, 2006). It is important to remember that the setting of standards is fraught with challenges due to resistance, fragmentation and legitimacy issues in collective action (Garud, Jain, and Kumaraswamy, 2002). The evolution towards modular systems can alleviate certain difficulties and facilitate contributions on various levels: setting standards as well as material contributions to collaborative innovation (Baldwin and Clark, 2006; Brusoni and Prencipe, 2006). The way the standards are set, that is the making of the design rules, tells a story of how a system evolved

[43] I'm grateful to Carliss Baldwin for suggesting this typology in her recent talk at the Strategic Management Society annual meeting in Miami on Nov 6, 2011.

(Brusoni and Prencipe, 2006) but also how it can be extended: and it is to the degree that it is modular that new additions can be fitted, holes filled, and features added[44]. The additions are uncertain and the knowledge added cannot be known in advance. This uncertainty characterizes open standards over markets for technology.

Whereas on markets for technology transactions involve clearly defined knowledge, open standards create a platform for knowledge in creation and lend themselves to collaborative innovation in early stages of knowledge creation. This can mean that companies successfully integrate users into their core business processes when developing innovations by building interfaces so that innovative users can centrally influence product development in a process of co-creation (Hienerth, Keinz, and Lettl, 2011). The companies these authors studied (Lego, IBM, and Coloplast) all asked for a transfer of intellectual property rights in order to exploit the ideas commercially. They used non-monetary incentives to reward contributors and they allowed users to become entrepreneurs building products issuing from collaborative innovation (Hienerth et al., 2011).

Open source as the third type of open system refers to the full disclosure of the underlying technology (the source code in software) and the permission to use, modify, and redistribute the technology[45]. The idea to keep software freely accessible in its source code takes its origin with software development practices in the 1970ies and Richard Stallman who founded the Free Software Foundation in 1985 (Levy, 1984; Moody, 2001)[46]. A more detailed account of the history of Free software can be found in section 3 under "Carrots and Rainbows".

The main difference between open standards and open source is that in open source the entire technology required to run the system represents a public good: it is freely available online and non-rivalrous as a knowledge good. The openness of open standards may be limited to interface documentations and specific layers (parts) of the technology needed to run a system whereas other critical parts are proprietary and belong to firms participating in promoting and developing a technology platform. However, the two types of systems outlined here overlap: Joel West (2003) has shown that hybrid platform strategies can involve both open source and proprietary components (see also Stuermer, Spaeth, and von Krogh, 2009).

The openness of the technology base had been an important argument for studying collaborative innovation in the example of open source software development (von Hippel and von Krogh, 2003). By following the artifact, that is the code, we could observe individual developers' work and contribution to the code base as well as ask for their rationale when reusing code from another project. Given the extensive history in the literature on software development on reuse studies in firms, the first study in this section on collaborative innovation across organizations in open source could compare and report on findings that appeared as surprising: frequent and substantial reuse despite a lack of formal programs for reuse and incentives by management. The surprise held only when the reader was familiar with the experiences from software development in firms as reference for open source software development. If that was not the case and the two contexts of software development were considered too different to allow for comparison, then research designs would have to be adapted.

The rationale from developers and the insights gained in studying code reuse contributed to a better understanding of collaborative innovation that involved long-term interactions among developers who coordinated their (mostly voluntary) work. The openness of systems, particularly the full openness of open source software, helped in attracting and retaining essentially non-contractible development resources because developers could search and identify useful components that advanced their projects and allowed them to work on what they preferred to do. Thus, the same openness that allows developers to identify each other around a shared goal or interest enables research on collaboration to proceed by following the artifacts that developers exchange.

---

[44] I owe the clarity of this idea to Carliss Baldwin (in her talk at the Strategic Management Society annual meeting in Miami on Nov 6, 2011).

[45] For more detail on the definition of open source turn to the website of the Open Source Initiative: http://www.opensource.org/

[46] For further information about Free software and the history of the GNU project turn to: http://www.fsf.org/about and http://www.gnu.org/philosophy/philosophy.html

Open source software development lends itself to study ever more detailed and specialized questions in innovation and our further and future projects document a part of such an agenda, see the last section of this thesis. The authors' experiences with the phenomenon of open source software development as a phenomenon in organization and management science led to a broader reflection on open source and a methodological contribution on studying phenomena (von Krogh, Rossi Lamastra, and Haefliger, 2011). The question whether readers follow a transfer of concepts from established literature (software development) to a novel context (open source) in order to test relationships thought to hold in well-understood contexts leads to a discussion of dimensions broadly thought of as similar or dissimilar. This question is highly relevant for future studies of collaborative innovation because the concepts that can and should be studied become more and more refined and of broader applicability in organizations as firms become more acquainted with open source software development communities and community members appreciate corporate support on various levels (Stewart, Ammeter, and Maruping, 2006). The inclusion and consideration of open source software in the design of information systems is one approach between firms and open source software communities, and the second essay in this section develops a set of questions for future research. The last section below picks up further issues in collaborative innovation with a focus on technology development.

CODE REUSE IN OPEN SOURCE SOFTWARE

Stefan Haefliger
Georg von Krogh
Sebastian Spaeth

Chair of Strategic Management and Innovation
Department of Management, Technology, and Economics (MTEC)
ETH Zürich, Switzerland

Published in Management Science vol. 54, issue 1, 2008 pages 180-193

Code reuse is a form of knowledge reuse in software development that is fundamental to innovation in many fields. However, to date there has been no systematic investigation of code reuse in open source software projects. This study uses quantitative and qualitative data gathered from a sample of six open source software projects to explore two sets of research questions derived from the literature on software reuse in firms and open source software development. We find that code reuse is extensive across the sample and that open source software developers, much like developers in firms, apply tools that lower their search costs for knowledge and code, assess the quality of software components, and have incentives to reuse code. Open source software developers reuse code because they want to integrate functionality quickly, because they want to write preferred code, because they operate under limited resources in terms of time and skills, and because they can mitigate development costs through code reuse.

INTRODUCTION

The particular context of open source software development[47], its organization in worldwide informal and virtual communities as it is Internet-based, the mostly public and archived communication between developers, and the availability of the code base have contributed to the general interest of researchers from many fields. The extraordinary success of some of the resulting software products (such as GNU/Linux, Apache, Bind DNS server, OpenOffice, Mailman) has drawn attention from the public and both software-creating and software-using organizations to this way of developing software. Yet, not all open source projects produce software targeted directly at the end-user. Some software is designed to be reused and to provide functionality to other software projects. For example, Lame, a music encoder, cannot be used directly but has to be built into, and used by, another program to create mp3 files. This leads to the question: do open source software developers tend to build software from scratch or do they rather reuse readily available knowledge and software code from other projects? On the one hand, free and open source software licenses, such as the GNU General Public License (GPL), grant permission to reuse software components within the limits of the license and one should expect open source software developers to build on each other's work. On the other hand, the many barriers to code reuse discovered in firms (Lynex and Layzell, 1998) raise doubts about the actual reuse behavior of developers in the absence of corporate reuse programs. Motivated by these two contradictory premises, this paper derives two sets of research questions from the literature on code reuse in firms and on open source software development and explore them using qualitative and quantitative data from 6 projects that vary in project agenda, age, size, and other factors.

Based on the premise that software development and code reuse in particular, hinges on technical as well as non-technical issues (Kim and Stohr, 1998), our analytical starting point is the behavior of individual developers. The general interest of this paper is to explore the practices of knowledge, and, in particular, code reuse in open source software development; what is being reused, when is it reused, by whom, and for what reasons.

The paper is organized as follows. Section 2 briefly discusses relevant theory and research on knowledge and code reuse in innovation and software development and identifies the research gap in the area of open source software development. We formulate research questions from the literature that relate to code reuse in open source software development. Section 3 gives an overview of the research method and the sample of projects studied. Section 4 presents the findings in the form of an inventory of code reuse across the project sample, and we complement these with findings regarding other types of knowledge reuse. This section also explores the research questions developed in Section 2. Finally, Section 5 concludes the paper and discusses implications of the study for management practice and proposes future research topics founded on this work.

[47] For better readability, the term open source will be used through- out this article, but the study also refers to Libre and Free soft- ware, which shares the same technical definition, but is driven by philosophical/moral considerations on freedom rather than techni- cal arguments: see http://www.gnu.org/philosophy/free-sw.html.

RESEARCH GAP: OPEN SOURCE SOFTWARE, KNOWLEDGE, AND CODE REUSE

Open source software development is an example of "private-collective" innovation (von Hippel and von Krogh, 2003): software developers derive private benefits from writing software and sharing their code and collectively contribute to the development of software. Such private benefits include enjoyment, fun, learning, reputation, and community membership (Lakhani and Wolf, 2005; Hertel, et al. 2003). An assertion in the private-collective view of open source software innovation is that benefits gained from contributing to the public good must outweigh the privately incurred cost of contributing to the software development (von Krogh and von Hippel, 2006).

The literature on technological innovation argues that knowledge reuse is an important mitigating factor for the cost of innovation (Langlois, 1999). Returns on investment in the creation of new knowledge hinges on the extent to which this knowledge can be applied across the development of new processes and products. Therefore, one of the central problems in the management of innovation is if and how firms reuse previously created knowledge across the various stages of an innovation process (see Argote, 1999; Majchrzak, Cooper, and Neece, 2004; Zander and Kogut, 1995).

The practice of knowledge reuse has been particularly relevant for innovation in the software industry and it is here that many of the most significant advances in the research on knowledge reuse have been made (Cusumano, 1991; Markus, 2001). While software code is notably explicit knowledge that is both readable by humans and enables a computer to perform specific functions, knowledge reuse may cover more than code reuse (Knight and Dunn, 1998: p. 295). As Barnes and Bollinger (1991, p.14) suggest: "The defining characteristic of good reuse is not the reuse of software per se, but the reuse of human problem-solving." Several types of knowledge can be reused across the different stages of software development (Frakes and Isoda, 1994): problem description, artifacts, project proposals, feasibility reports, enterprise models, data dictionaries, prototypes, decision tables, pseudo-code, source code, databases, the tacit knowledge of developers, networks of developers, and so on (for an overview, see Cybulski et al, 1998; Prieto-Diaz, 1993; Ravichandran, 1999). The documentation of software design patterns facilitates the reuse of problem solving in software engineering, particularly when using object-oriented languages (Gamma et al., 1995; Schmidt, et al 1996). The reuse of software is enabled through modular software architectures and the development of generic software components. However, the design of generic components requires substantial investment for a firm that can only pay off in the long run if and when the firm saves development costs through component reuse in software projects (Banker and Kauffman, 1991). In the software industry, firms that reuse code on more than one project can amortize development costs faster and reduce development time in new projects (Barnes et al., 1987; Banker and Kaufmann, 1991). Reusing code and components from software libraries also enhances the quality of new software products by allowing for fully tested and debugged software (Knight and Dunn, 1998).

In spite of the reported benefits, several studies on software development firms have found that code reuse in software development is problematic and that the success of corporate reuse programs hinges on organizational factors more than on technical factors (Apte et al., 1990; Isoda, 1995; Kim and Stohr, 1998; Rothenberger et al., 2003). This literature also provides insights regarding the possibilities of code reuse in open source software: in software development firms, corporate reuse programs need to commit an initial investment to reuse (Isoda, 1995) in order to generate long-term savings including life-cycle benefits such as maintenance (Banker et al., 1993; Basili, 1990). Program success depends on standards and tools provided to developers (Lim, 1994; Kim and Stohr, 1998), on the certification of software (Knight and Dunn, 1998), as well as on the incentives for developers to reuse (Poulin, 1995).

Systematic reuse in software development firms requires years of investment (Frakes and Isoda, 1994) in order to create and maintain reusable code and other knowledge (Lim, 1994), populate repositories and libraries (Griss, 1993; Poulin, 1995), and provide tools for developers to identify and reuse code (Isakowitz and Kauffman, 1996). The organization's funding structure usually needs adaptation to coordinate reuse investments across the organization (Lynex and Layzell, 1998), because developers need to work in reposi-

tories and components that are not directly linked to a product. Pilot programs, accompanied by code reuse performance metrics (Frakes and Terry, 1996), may instigate systematic reuse that can be monitored across the organization (Banker et al., 1993). Management needs to appoint champions as sponsors, reuse-librarians or -coordinators (Isakowitz and Kauffman, 1996; Joos, 1994; Kim and Stohr, 1998).

The success of a corporate reuse program depends on whether the costs to the developer of search and integration are lower than the costs of writing the software from scratch (Banker et al., 1993). According to the literature, this can be achieved by creating standards and tools that facilitate the search for and integration of software components. Elaborate classification schemes (Isakowitz and Kauffman, 1996) facilitate the use of and access to libraries and lower search costs for developers. Domain analyses, documentation, and quality standards enhance the ability to reuse software components (Poulin, 1995). Ideally, the information accompanying reusable code should incorporate a quality rating or certification in order to enhance the developer's trust in code and components written by someone else (Knight and Dunn, 1998; Poulin, 1995). This emphasis on quality stems from the software developer "(feeling) that defects in a reused code could have a substantial negative impact on whatever system he or she is building" (Knight and Dunn, 1998: 293).

Incentives play a crucial role in corporate reuse programs (Isoda, 1995; Lynex and Layzell, 1998; Poulin, 1995; Tracz, 1995). The monetary or reputation-based incentives offered by software development firms (Poulin, 1995) need to outweigh the dominant notion that code reuse is "boring" or "less satisfying" than writing code (McClure, 2001; Tracz, 1995), overcome the not-invented-here syndrome, and the general resistance to change in organizations (Lynex and Layzell, 1998).

In contrast to software development firms, open source software development projects do not feature corporate reuse programs and usually have no financial resources to invest in tools, standards, and incentives. This could have adverse effects on code reuse; for instance, the lack of incentives could prevent systematic code reuse by open source software developers who are known to code for the creative challenge and the fun of tackling "technically sweet" problems (see Raymond, 2000: 25; Lakhani and Wolf, 2005; Hertel et al., 2003). Thus, in the absence of corporate reuse programs, it can be reasoned that open source software development projects would need "equivalent" mechanisms to substitute such programs. Based on the review of the literature on code reuse in software development firms, the following questions can be formulated regarding such mechanisms:

Research question 1 a: Do equivalents to standards and tools (found in software development firms) support code reuse in open source software development?

Research question 1 b: Do equivalents to quality ratings and certificates (found in software development firms) support code reuse in open source software development?

Research question 1 c: Do equivalents to incentives (found in software developing firms) support code reuse in open source software development?

Despite the absence of corporate reuse programs, research on open source software development provides reasons to expect systematic code reuse among developers. Three leading reasons are examined in this study. First, the demanding requirements for the functionality and architecture of the code after the inception of an open source project might make it rational for developers to reuse already existing code. Second, the desire to work on preferred tasks should lead developers to reuse code that they prefer not to write on their own. Third, resource constraints in terms of time and skills should lead to reuse behavior in open source software development.

The existence of a code base seems crucial in order to mobilize open source developers, as shown for example in a study of the Freenet project (von Krogh et al., 2003). After a project's inception, its developers have to fulfill what we call a "credible promise," best defined by using a quote from Lerner and Tirole (2002: 220): "a critical mass of code to which the programming community can react. Enough work must be done to show that the project is doable and has merit." The credible promise enables sufficient functionality of the software to catch the interest of potential users and developers. A side effect of reusing components is its impact on the software architecture which is also evaluated by prospective developers (Baldwin and Clark, 2006). The reuse of a software component takes advantage of a design option (Baldwin and Clark, 2006; Favaro et al., 1998, MacCormack et al., 2006) and adds to the modularity of the overall architecture. Given the advantages of modularity in software development across the time span of the

project (Baldwin and Clark, 2000; Garud and Kumaraswamy, 1995), the reuse of components seems rational at any time, not only during the early phase of a project.

Developers of open source software are known to self-assign to tasks based on their preferences and ability (Benkler, 2002; Bonaccorsi and Rossi, 2003; Yamauchi, et al., 2000) and they seek a creative challenge and fun when writing software (Lakhani and Wolf, 2005; Hertel et al., 2003). Yet, some essential tasks in open source software development are considered to be mundane and boring (Shah, 2006; von Hippel and Lakhani, 2003). A solution to achieving an operational software product could be code reuse. Developers who face several essential tasks may choose to solve the less preferred ones by reusing code rather than writing everything from scratch.

Any open source software developer can consider the vast amount of available open source software when building their own code base. Open source licenses convey the basic rights to the developer to retrieve the code, inspect and modify it, and to freely redistribute modified or unmodified versions of the software to others. Such a license inherently encourages a developer to reuse code, although a license might require that derivative works are released under an open source license as well.[48] The cost of contributing to open source software development can be substantial. For example, those who want to join a community of developers must demonstrate considerable skill at solving technical problems. Von Krogh, Spaeth, and Lakhani (2003) found that newcomers to a project reused software they had written for other projects in order to make their first contributions. Contributions to the public good incur private costs to developers (von Hippel and von Krogh, 2003). Hence, one should expect that developers find it opportune to mitigate their development costs through code reuse from other projects. Empirical studies of Apache and Linux developers have demonstrated that a strong incentive for developing open source software is to solve a technical problem by writing software code and getting feedback from other users (von Hippel, 2001; Hertel et al. 2003). If software that solves the problem is already available under an open source license, there is no reason why a developer should write their own code. This economic logic should apply beyond the initial release of the software. Hence, the following questions can be formulated:

Research question 2 a: Does the open source software developers' aim to publish workable software as early as possible (credible promise) supports code reuse in open source software development?

Research question 2 b: Does the open source software developers' self-assignment to tasks according to their preferences and ability supports code reuse in open source software development?

Research question 2 c: Do Open source software developers' resource constraints (in terms of limited time and skills) supports code reuse in open source software development?

In sum, knowledge and code reuse are fundamental to the economics of innovation and central to software development. The characteristics of open source software development could provide both favorable and inauspicious conditions for code reuse but, to date, there is no empirical research available on the topic.

[48] The exact definition of what poses a derivative work is disputed. There are, for example, many ways in which programs can interact. What kind of interaction implies a derivative work, versus a mere aggregation of individual programs, is a controversial issue among the various factions of producers and consumers of open source software.

RESEARCH METHOD AND SAMPLE

This section describes the sample selection process and the research method that guided this study. The literature review on code reuse and open source software led to the formulation of precise research questions for code reuse. The available literature indicated reasons why code reuse might occur in open source software development without providing empirical evidence. Thus, we proceeded to explore the questions using a multiple case study design drawing upon several different data sources (Yin, 1989). An open invitation to a short, anonymous, web-based survey was posted on a developer mailing list in order to decide whether the topic was of any interest and relevance to the field. The resulting 30 replies indicated that knowledge and code reuse are important issues and an integral part of open source software development practice. Next, interviews were conducted and emails, public documents, and code were gathered from an initial sample of 15 projects. The sample included a wide variety of software products such as office software, games, a hardware driver, and an instant messenger client. The projects needed to fulfill three conditions to be included in the sample: 1) the project was under active development, allowing us to track its development activity and interview key developers, 2) the source code modifications of the project needed to be available online, and 3) the project had to have been in existence for at least a year which enabled us to track code reuse over time.

For a more in-depth analysis, the initial sample was reduced to a "core sample" of 6 projects exhibiting variance on the sampling criteria: size (lines of code (LOC), number of developers), objective, date of inception, target audience for software product, license, and programming language. By keeping the high variety of project characteristics, a sampling bias was avoided (e.g., Stake, 1995). Moreover, in order for projects to be included in the core sample, their core developers needed to be available for interviews. The resulting core sample is presented in Table 12.

Project	Objective	Lines of code	Inception	Developers*	Target audiences	Licenses**	Programming languages
Xfce4	Xfce is a lightweight desktop environment for Unix-like operating systems. It aims to be fast and lightweight, while still being visually appealing and easy to use. Xfce 4 is a complete rewrite of the previous version. It's based on the GTK+ toolkit version 2.	2,435,172	2001	17	End-user	BSD, GPL	C
TikiWiki	Tiki CMS/Groupware (aka TikiWiki) is a powerful open source content management system (CMS) and groupware that can be used to create all sorts of Web applications, sites, portals, intranets, and extranets. TikiWiki also works great as a Web-based collaboration tool. It is designed to be international, clean, and extensible.	842, 025	2002	89	End-user/ developer	LGPL	Javascript, PHP
Abiword	AbiWord is a free word-processing program similar to Microsoft® Word. It is suitable for typing papers, letters, reports, memos, and so forth.	1,368,264	1998	63	End-user	GPL	C, C++, Objective C
GNUnet	GNUnet is a framework for secure peer-to-peer networking that does not use any centralized or otherwise trusted services. A first service implemented on top of the networking layer allows anonymous censorship-resistant file sharing. GNUnet uses a simple, excess-based economic model to allocate resources. Peers in GNUnet monitor each others' behavior with respect to resource usage; peers that contribute to the network are rewarded with better service.	259,586	2001	16	End-user/ developer	GPL	C
Irate	iRATE radio is a collaborative filtering system for music. You rate the tracks it downloads and the server uses your ratings and other people's to guess what you'll like. The tracks are downloaded from websites that allow free and legal downloads of their music.	109,201	2003	21	End-user	GPL	Java
OpenSSL	The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and open source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library.	1,014,816	1999	12	Developer	Apache-style	C

*Number of active or core developers as stated by the project.
**For details on licenses, see, e.g., http://opensource.org/licenses.

Table 11: Core sample overview

The data sources from the core sample included interviews, source code, code modification comments, mailing lists, and various web pages related to the projects. Between December 2003 and June 2004, interviews were conducted with 21 core developers[49] of sample projects, 12 of

[49] A "core developer" has CVS access, contributes the bulk of the code, and assumes administrative tasks. See also von Krogh et al. (2003) and Shah (2006).

which belonged to the core sample.[50] The interviews were semi-structured, conducted by telephone, and lasted on average 50 minutes. The developers were contacted by email first. Two thirds of the interviews requested were carried out. The interviewees of the core sample consisted of members of the inner circle of the current development team. We were able to interview the developers of 74% of all instances of initial software component code imports into the core sample projects (referred to as "architectural reuse," see Section 4). In order to protect their privacy, the names of the respondents are replaced by capital letters throughout this text. Developer interviews were used to increase familiarity with the project, clarify open issues, and to examine the motivation for knowledge and code reuse.

The developer mailing lists of all core sample projects were analyzed two weeks prior to and after the first reuse of a component. We coded all reuse-related comments and measured the length of discussions (by anyone on the lists) spurred by the reuse incidents.

The source code of the core sample was initially available on project websites and managed using the Concurrent Versions System (CVS) source code management tool (for a description, see von Krogh et al. 2003). The CVS source code repositories of the core sample were retrieved and stored in a local database in order to enable the analysis of source code changes and the associated comments in the CVS.[51] The source code was examined in four ways: accredited lines of code reuse, identification of reused components, identification of functions within the components and their reuse across the core sample, and an authorship analysis.

First, a rough analysis was performed, including how many lines of software code were directly 'copied and pasted' from other projects. In order to find these, the originating project and/or authors needed to be accredited in the CVS comment and, therefore, they were referred to as "accredited lines of code-" reuse. Developers commented and accredited these lines, such as the following comment made by developer G: "added configuration file parsing without OpenSSL using code from xawtv." Based on this analysis,[52] 38,245 accredited lines of code were identified across the core sample, - a relatively low value compared to more than 6 million lines of code in the core sample projects. The developers commented that copying lines of code only occurred infrequently and in small quantities, but that giving credit was mandatory. However, there might still be an unknown quantity of imported lines of code which was not explicitly accredited.

The identification of reuse of software components was done in an automated manner by filtering the source code for programming statements used to include components.[53] In the next step, the functions within each reused component were identified and a more fine-grained analysis identified the reuse of functions offered by the components identified across the core sample. The tool "Doxygen[54]" was used to extract the software architecture, specifically the application programming interface (API) of the identified components. This XML file-based information was used to search all code modifications of the six sample projects for function calls added to each project's software, using functionality provided by the included components. This result

[50] For four of the six projects, two developers were interviewed. For GNUnet only the founder (who wrote the bulk of the project's code) was interviewed, and for xfce4, three developers were interviewed.

[51] All projects were recorded from their inception until mid-2004. An exception was Xfce4, which is a rewrite of Xfce3. Here, the inception date was adjusted to the time when development actually picked up, and the developers migrated to working on Xfce4. This meant relocating 268 out of 62,000 CVS incidents to the new inception date.

[52] To allow for the identification of accredited lines of code (ALOC), we applied a Bayesian filter. This is based on an algorithm that estimates the probability of a code modification to be a knowledge reuse incident, using conditional probabilities, by rating the occur- rence of specific words based on training data. In this study, the filter was trained by manually analyzing the source code modifi- cations of two projects. We were thus able to calculate the proba- bility that a source code modification comment related to imported lines of code from another project. Resulting hits and probable hits were examined manually in order to settle whether or not they represented ALOC reuse. As a reviewer correctly pointed out, this method provides the lower bounds for ALOC reuse as the filter might miss actual ALOCs. For the Bayesian filter reference, go to: http://www.paulgraham.com/spam.html.

[53] The search included source/header files through statements such as"#include"forC/C++,"package"/"import"forJava,and" require"/ "require_once"/"include"/"include_once" for php-based projects. Sys- tem files, such as files belonging to the standard C library or the Linux kernel, were filtered out.

[54] Doxygen is described as "a documentation system for C++, C, Java, Objective-C, IDL (Corba and Microsoft flavors) and to some extent PHP, C#, and D." It is publicly available at http:// doxygen.org.

Component	Component description	Reuse date	Reuses	
			Architectural	Functional
Abiword				
libpng	Display "png" graphics	1999-02-16	1	26
glib-gtk-gdk	Graphical toolkit	1998-07-28	24	542
fribidi	Display fonts (bidirectional)	2001-09-13	5	26
zlib	Compression utility	1999-02-16	1	10
Libxml	xml file parsing	2000-07-27	3	20
enchant	Spell checking library wrapper	2003-07-13	1	11
ispell	Spell checker	1998-12-28	2	18
Libwv/libmswordview	View MS Word files	1998-08-28	9	180
libiconv	Unicode	2000-02-01	1	2
expat	xml parser	1998-08-31	1	14
libglade	Graphical toolkit construction	2000-10-11	Indirect through gtk	5
libpopt	Parsing command line options	1999-03-11	2	11
libfreetype	Display fonts	2002-05-12	5	8
libwmf	Display "wmf" graphics	2001-09-28	3	5
libjpeg	Display "jpg" graphics	2001-04-26	1	7
libcurl	Download files	2002-04-30	2	4
libgal	Gnome accessibility functions	2000-12-16	6	34
pango	Display internationalized text	2002-05-05	15	126
GNUnet				
openssl	Encryption	2001-09-05	10	28
libgcrypt	Alternative encryption lib	2003-01-15	8	219
libpopt	Parsing command line options	2002-01-06	1	3
gdbm	Data base	2002-05-05	1	5
tdb	Alternative data base	2002-05-14	1	8
MySQL	Alternative data base	2003-03-01	1	15
gtk/gdk/glib	Graphical toolkit	2002-03-09	1	160
libglade	Graphical toolkit construction	2003-10-28	Indirect through gtk	4
zlib	Compression utility	2001-08-30	1	2
sleepycat db	Alternative data base	2003-03-30	1	1
irate				
javalayer	Mp3 player	2003-03-26	1	54
xerces	xml parser	2003-03-26	3	58
nanoxml	xml parser	2003-05-26	1	9
swt libraries	Graphical toolkit	2003-07-24	3	11
httpClient	Download files	2003-08-06	2	3
libmadplayer	Mp3 player	2003-03-30	3	3
OpenSSL				
zlib	Compression utility	2000-11-30	1	5
TikiWiki				
Pear::DB	MySQL wrapper for php language	2003-04-09	1	N/A (MySQL)
MySQL	Data base	2002-10-08	Via Pear::DB	15
Smarty	Php template engine	2002-10-08	1	38
hawhaw	Provide "wap" version	2003-05-15	1	6
htmlarea	Interactive html editor	2003-04-23	2	2
JGraphPad	Java applet paint program	2003-04-24	1	1
overlib	Display browser tooltips	2002-11-25	1	1
jscalendar	Display calender in Web browser	2002-10-08	1	1
adodb	Data base abstraction layer	2003-07-15	3	5
GraphViz	Generating graph layouts	2003-04-02	1	1
php-pdf	Convert files into pdf files	2002-10-08	2	18
phplayers	Php menu system	2003-11-25	1	1
wollabot	Generic IRC chat bot	2003-11-15	1	1
xfce4				
gtk/gdk/glib	Graphical toolkit	2001-02-14	37	863
libiconv	Font conversion to/from unicode	2002-12-15	1	1
libxml2	xml file parsing	2003-02-04	4	21
alsa	Linux sound library	2003-04-26	4	43
oroborus	Window manager	2002-05-03	6	63
libwnck	Window Navigator Construction Kit	2002-10-18	10	40
pango	Display internationalized text	2002-05-03	1	17
Sum:			200	2,775

Table 12: Component reuse inventory in the core sample

covered high-level calls using the public API of the components.

Using the first occurrence of each included file or reused function, the analysis identified 2,975 unique reuse incidents. Of these incidents, 200 imported a component (or part of a component) and 2,775 incidents made use of the functions offered by the reused components. A total of 55 reused components were identified in this way, leading to a component reuse inventory (as shown in Table 13). The identified list of software components was sent back to the project lead developers (listed on the projects' web pages) in order to check its validity. Out of 6 projects, 4 replies were received validating the results. One respondent confirmed in the interview that the project only reuses one optional component. In one case, the lead developer was too busy to validate the findings.

Finally, the timing of component reuse incidents and statistics on the developers who performed the reuse were collected for all component and function reuses in the component reuse inventory. In the next section, we turn to the findings.

FINDINGS

The aim of this section is to shed light on the research questions developed in Section 2. While we mainly report on code reuse, we also sustain the notion that knowledge reuse in software development covers reuse of problem-solving as well as code (Barnes and Bollinger, 1991, Section 2) and, thus, include other forms of knowledge reuse where appropriate. First, an inventory of the projects studied shows that developers predominantly reuse software components. Second, the research questions are explored and contrasted with the analysis of component reuse and the interview data in order to answer why reuse happens.

4.1 Knowledge and code reuse

There were three broad forms of knowledge and code reuse in the core sample: algorithms and methods, single lines of code, and components. First, an algorithm is a finite set of well-defined instructions for accomplishing some task or solving some problem which, given an initial state, will result in a corresponding recognizable end-state (adapted from wikipedia.org, 2004). Methods contain several alternative algorithms and other scripts for solving a problem, rules for choosing between them, and they can be expanded to cover a large problem area. The reuse of algorithms and methods includes the examination of source code or other information, but also the interpretation and adaptation of cues about technical problems and their solutions, abstraction, and implementation in a local context. Nearly all of the developers interviewed mentioned the reuse of algorithms and methods in their open source software development. The analysis across the core sample revealed that knowledge reuse in the form of methods and algorithms is frequent but rarely credited. Interviews revealed that developers spend non-negligible amounts of time studying scientific publications (such as engineering journals) and standard specifications, or learning from the source code (and its documentation) of related projects. The reuse of algorithms and methods not accompanied by software code reuse is impossible to document completely, because it is part of the individual's learning and usually not made explicit.

Second, copying specific lines of code from external projects is a systematic and direct form of code reuse in open source software development. The analysis of imported "accredited lines of code" amounted to 38,245 across the core sample. For reasons described in Section 3, this form of reuse was relatively rare, hard to quantify, and not further pursued in the current study.

Third, all the core sample projects reused software components. A software component is a software technology for encapsulating software functionality, often in the form of objects, adhering to some interface description and providing an API, so that the component may exist autonomously from other components on a computer. Technically, this autonomy allows the developer to treat the component as a "black box." The components were either integrated into the code of the project or linked to it. Linking a component to the software could happen either at the time of compilation (static linking) or at run-time (dynamic linking). Reuse (or acquisition) of components can be "black-box" or "white-box" (Ravichandran and Rothenberger, 2003), depending on whether changes were made to the reused code. With very few exceptions, the reuse in this sample amounted to black-box reuse since the components were reused without modifications. Across the core sample, 55 components were reused, representing a total of 16.9 million component LOC, while the total LOC of the core sample was 6.0 million (not including the reused LOC). A complete list of reused components can be found in Table 2. Each identified component reuse incident is listed together with a minimal description and its date of reuse. All the components reused in the core sample are maintained as external projects which means that they are available through a dedicated project website, provide code releases or open development, or all of the above.

A closer analysis of the components revealed two distinct types of code reuse: architectural reuse and functional reuse. In order to make use of components that are not developed inside the project's community of developers, a developer has to first search for and integrate a suitable component. The decision to reuse a component introduces an architectural change to the software because it changes its overall structure (Baldwin and Clark, 2000; Ulrich, 1995). According to the developers (B, H, M, L), the decision to reuse a component is based on the functions this component offers. In order to make the code

reuse decision, the developers studied not only the software code but also the documentation accompanying the code, web pages with frequently asked questions (FAQs), overview documents and similar web-based resources before deciding to reuse a component. The first step of reuse is then the inclusion of the component in the software. The core sample contained 200 instances of architectural reuse which import a component or part of a component. In Table 2, they are attributed to each of the 55 reused components in the fourth column.

The second type of component reuse was termed functional and based on previous architectural reuse. Each component offers a number of functions that may or may not be used by the originating program. Through architectural reuse of a component, the developer makes functionality available to the program. The actual use of some of these functions executes the options inherent in the component. Functional reuse incidents became visible through fine-grained analysis of the code base of the sample that revealed each available function. Only the first call of a new function was recorded in order to determine whether it was using functionality provided by components. The core sample contained 2,775 functional reuse incidents. The functional reuse incidents correspond to a reused component and are listed in the fifth column of Table 2.

4.2 Substituting the corporate reuse program

The component reuse inventory demonstrates that open source software developers routinely and widely reuse software components across the sample. The following analysis explores questions 1a through 1c in order to explore how the context of open source software provides equivalent mechanisms that corporate reuse programs feature, namely lower search costs, establishment of quality standards, and the provision of incentives.

The perceived costs to a developer of searching and integrating a new component must stay below the effort of writing software from scratch (Banker et al., 1993). Tools and standards facilitate the search and integration of existing components (Ravichandran, 1999), and reuse in open source software development should only be expected if equivalents to corporate tools and standards exist. Internal search repositories, a solution proposed by Banker et al. (1993), could not be found in the core sample. However, a few large repositories of open source software projects (Sourceforge.net, Savanna, Berlios, etc.) as well as dedicated index and search tools (e.g., Freshmeat, koders.com) offer free infrastructure for projects of various domains and target both end-users as well as developers. Distributions, such as Debian GNU/Linux, publish extensive information that helps developers identify components and dependencies of components which they contemplate using . We found that 85% of all reused components in the core sample were listed in the Debian package repository.

However, even more important than repositories and search engines were means of local search in a known space (March, 1991). Several developers (D, K, F, H) underscored the importance of their respective software community for finding relevant knowledge and code. One developer (F) suggested that of all sources, his project's developers and mailing list participants were the most direct and efficient source of information about reusable knowledge and code. Developer (H) suggested:

> *"I'll post on the mailing list, - the developer list. I just ask everybody: does anyone know about this? Can anyone recommend a good library? Chances are somebody uses one. Might not even be writing code, but they might know something about it."*

Next, standards provided a means of lowering search and integration costs. In the core sample, standards involved stable and documented interfaces to the component functionality, as well as its accessibility in the project programming language, and influenced the developers' decision to reuse. By using a well defined and designed set of variables and commands (API), the developers could access the component's functionality, thus reducing the effort to understand the component (Developers A, G, H, D). With a good interface available, the developers did not have to fully understand the inner technical workings of the component in order to be able to use it and they could integrate the component into the software more easily. Accessibility of the component to the project's programming language also proved to be important. For example, the developers of iRATE radio considered replacing a component with another external library (libmadplayer) in order to take advantage of its advanced functionality. However, a lack of compatibility between

the two programming languages Java and C made this difficult:

> *"But if we switch to libmad[player], we'll have to maintain the Java interface from libmad to iRATE. [...] They don't have a Java interface right now. Basically it's to use libmad, it's written in C, and [it is] sort of difficult to interface C with Java code. Whereas [for] Madplay, - it's just a program [front end for libmadplayer] and Java doesn't really care what program it is as long as it can run from command line. But [with] libmad, we'll actually have to call internal routines, - it's much more difficult to keep it up to date." (Developer I)*

The risk of unforeseen changes within reused components was mediated by what developers considered "a wide consensus" among open source software projects to guarantee API stability within major releases. As developer D states, the effort of integrating a component increases when the interfaces change too often:

> *"If [the component] changes constantly and it's incompatible with your project, it causes overhead. You don't want to have 50 million conditionals for every software that exists. In this case you might choose to copy over their code or email the software's maintainers and say; "listen, we're trying to use your code but you keep changing it on us, is there any way you can keep this stable?" Frequently, they will be happy to comply because they'll have extra users of their code that help find bugs. There are several options, it's good to know the policy but it's not necessary for the initial use of their code."*

The interviews revealed that open source software development offers equivalents to search tools and standards which facilitate the developers' search for and integration of components, positively replying to question 1a.

Knight and Dunn (1998) point out that developers in firms are unlikely to reuse a component unless they can trust its quality, since an external component can potentially harm the overall system. Therefore, they propose certifying components for reuse. In the core sample, the "popularity" of a component served as a substitute to certification and generally signaled the quality of the software (developers H, G, F). The developers reasoned that bug fixes in the software are more frequent in widely used components leading to better quality:

> *"There's a higher probability that more people have looked at the code, figured out if it works, how it works, and probably fixed more bugs if there are any. It's the whole peer review thing: the more people have looked at the code and still use it, the more it's trustable." (Developer F)*

A straightforward indication of the popularity of a component is its inclusion in a major software distribution such as Debian which is peer reviewed, actively maintained, and reaches a wide user base. (As mentioned above, 85% of the reused components in the inventory were included in the Debian distribution). This answers question 1b: by evaluating the popularity of components, open source software developers indeed use a proxy to certification and quality standards in order to support code reuse.

A software development firm needs to create incentives for developers to reuse code because they generally perceive reuse to be less rewarding than writing new code. Von Hippel and von Krogh (2003) argue that, in the private-collective model of innovation, developers expend resources privately to contribute to a public good. If the developers perceive their resources as limited, reuse could help to mitigate development costs. In this case, the net savings in development costs through reuse should act as individual incentives. Support was found for this conjecture: developers in the core sample explicitly reused code in order to reduce their costs of creating the software. They spent available resources in consideration of both available time and skills (developers K, D, H, A):

> *"The primary driver of that [reuse] decision is making the project the best it could be. The fact is that we're mortal and we don't have an infinite amount of time to rewrite everything. So even if the other project's code isn't perfect but good enough, you're simply going to use it because if you've got a thousand bugs to fix you don't want*

to spend the next year rewriting all the software you could just use from someone else." (Developer D)

The interviews confirmed the existence of resource constraints in open source software development. The economic rationale of saving development costs, where possible, was consistent among developers' replies. The need to advance the code base and approach the objectives of the overall project was weighed against the time and skills available and influenced the decision to reuse code, as shown in this example:

"There were cases where it was better to have a third-party program. OK, for example jgraphpad, which is a java applet, to have graphics. We wouldn't necessarily have the expertise on the team to do that, or the time or the interest, but that was a perfect match between what they were doing and what we were doing." (Developer K)

An additional incentive to reuse was the option to outsource the maintenance work through reuse. For 53 out of 55 reused components, at least one new release became available after the first date of reuse. Hence, the reusing projects could benefit from "free" maintenance by other projects. Developers (B, K, H) considered external maintenance as an incentive to reuse components because it lowered the long-term costs of producing a component inside the community by the effort to maintain it, particularly regarding internal bugs and errors. The developers in our sample systematically reused components because, first, they saved effort by not having to write the component, and second, by not having to maintain it in the future. These findings relate to question 1c; developers' limited time and skills create incentives to seek savings in development costs through code reuse. The pattern that an economic logic influences reuse behavior also positively replies to question 2c.

4.3 Fulfilling the credible promise

The credible promise, or the release of workable software that is complete enough to work on, helps attract users and potential developers to a project (as described in von Krogh et al., 2003). Accordingly, this section explores questions 2a through 2c. One way to quickly establish a working code base is to integrate existing components and building on existing functionality. The findings shed light on question 2a: developers reused components as early as of day one of the project's inception. Figure 7 shows that 666 of 2,975 reuse incidents (22%) already occurred during the first 10% of the observation period for the core sample projects. The credible promise is fulfilled in the core sample: the first public software release happened on average already 44.5 days after a project's inception.[55]

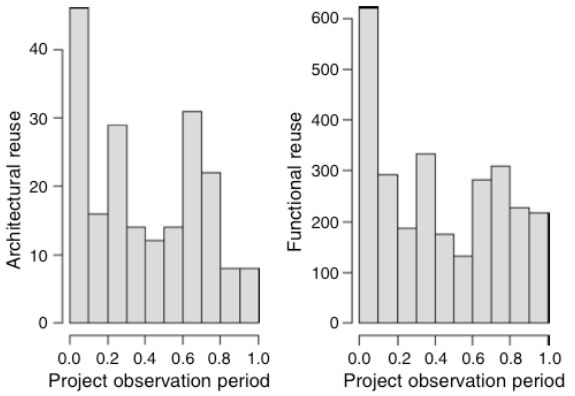
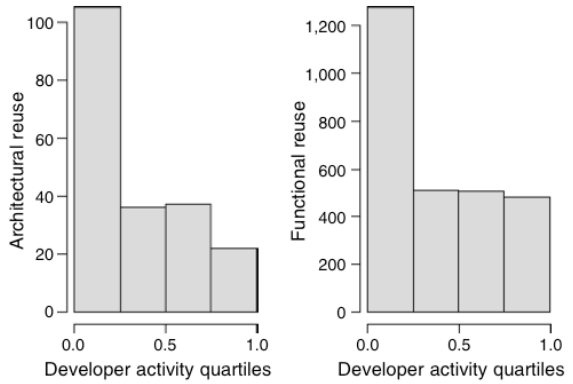


Figure 7: Reuse incidents over the total observation period

The patterns of architectural reuse resemble those of functional reuse; new components (or parts of components) were added to the code base throughout the observation period. Hence, developers make use of the (modular) design options by adding components. This observation is consistent with a claim made in the literature that developers exercise design options in software architecture (Baldwin and Clark, 2006).

As mentioned above, developers chose areas they like to work on through self-assignment of tasks. But since an open source software project also requires the execution of mundane and difficult work for developers, question 2b asks if code reuse could help to evade the writing of "mundane" code and focus on more rewarding programming tasks specific to the project and that fit the developers' skills.

[55] The day of inception is the first day on which code is added to the repository. It is possible that part of the source code existed earlier outside of the repository.



Notes. The developers' "life span," that is, their total coding activity over time, was divided into quartiles. The reuse incidents were allocated to the individual activity quartiles in order to visualize the reuse incidents over the developers' "life spans." The histogram shows when reuse incidents happened for every quartile. "0" on the activity axis indicates a reuse incident as the first activity (in LOC), "1" a reuse incident as the last observed LOC written by the developer.

Figure 8: Reuse incidents during the developers' life span

The developers' individual reuse behavior shows increased reuse early on in the developers' coding activity for the project (Figure 8). The total median in this sample (architectural and functional reuse combined) was 0.28 (avg: 0.37, sd: 0.32). Thus, on average, developers performed reuse after having contributed a third of their total number of lines of code.[56] This implies that most developers remained active after a reuse incident and continued to write code for the project. The interviewees confirmed the developers' preference for reusing early during their active period on the project. They (J, H, K, A, E) perceived reuse as an opportunity to get rid of mundane, time-consuming, or difficult coding tasks, that helped them to work on their preferred tasks. Developer E sums it up:

> *"Code reuse is just helping us to get the job done, so I can work on something that is more interesting."*

These findings answer question 2b: software reuse helped developers to get mundane or difficult tasks done and allowed them to focus on "interesting" (preferred) areas of work.

Finally, as elaborated in subsection 4.2, resource constraints were explicitly and frequently mentioned as reasons for code reuse by developers. The developers benefited from "free" maintenance and improvements made to project-external components and chose the least costly path to ensure that workable code could be released and progress was being made. This behavior relates to the finding that developers spend their scarce resources economically. Component reuse helped to advance the project, thus answering question 2c positively.

[56] The data also showed that long-term and more active developers performed more code reuse, not in relative, but in absolute terms. The reason may be that these developers have acquired a better familiarity with the code base. We are grateful to the associate editor for pointing this out. Dividing developers into two groups (contributing for more/less than 50% of the observation period pre- ceding the reuse incident), additional analysis showed no signifi- cant differences between the groups in code reuse frequency over a developer's life span in a project.

CONCLUSION AND IMPLICATIONS

In the "private-collective" innovation model, the benefits must outweigh the cost of contributing to the public good (von Hippel and von Krogh, 2003). Knowledge reuse can be a strategy to mitigate the costs of innovation (Langlois, 1999) and commercial software engineering practices emphasize the reduction of developments through code reuse (e.g., Barnes, et al. 1987; Barnes and Bollinger, 1991; Banker and Kaufmann, 1991). This study departs from two contradicting issues, namely that open source software licenses are designed to enable and encourage sharing and building on others' work, yet reuse is hard to achieve in commercial settings. It shows that developers in open source software projects actively reuse available code and other knowledge that solve their technical problems, and it presents empirical evidence on the extent of code reuse, and the development practices of developers in open source software projects where resources are scarce, highlighting the importance of reusable software components. In the sample of six open source projects, this research identified 55 reused components comprising 2,975 reuse incidents. The findings are presented against the back drop of the literature on code reuse in software development firms, offering insights with regard to the context of open source software. The data from the core sample showed that developers used tools and relied on standards when reusing components. They used "popularity" as a proxy for quality ratings and acted under resource constraints (time and skills) when selecting reusable components. While core developers participated in the reuse activities, they offered no explicit encouragement for code reuse across their project. As predicted by the existing literature on open source software development, building an initial credible promise required code reuse early on during the project, with developers continuing to reuse code as resource constraints persisted and suitable open source components were available. The developers continued to reuse and stay active after their initial, extensive reuse behavior. This finding is consistent with Shah (2006) who showed that the type of tasks tackled by long-term developers changes over time. In summary, the developers reused for three reasons: they wanted to integrate functionality quickly (first public release after 44.5 days on average), they preferred to write certain parts of the code over others, and they could mitigate their development costs through code reuse.

Implications for research

Most component reuse in open source software development crosses project boundaries, thereby enlarging the project resources by effectively outsourcing part of the development. In particular, these additional resources might help young projects to gain the necessary momentum to reach a critical mass. This represents an alternative strategy to community growth and resource mobilization (e.g., Bonaccorsi and Rossi, 2003). Future research should examine this form of growth in more detail.

The software reuse literature (Tracz, 1995) estimates that the cost of building reusable components adds up to 200% of additional development costs. Future research should uncover who carries these costs in open source software development and why. As this study has shown, repositories such as Sourceforge offer vast amounts of reusable software and most components were released by projects dedicated to that specific component development. Casual evidence from the core sample suggests two possible explanations: the developers' mobility across software projects (proprietary and open source) benefits from reusing parts of their own work in parallel or subsequent projects. Second, reputation rewards from peers for "clean", that is modular and well structured, code justify the private, additional efforts to build reusable components. Both arguments deserve further in-depth examination.

Certain projects reuse more code than others, but the antecedents of reuse in open source software development are largely unknown. Future research should identify individual and organizational characteristics that impact on reuse. Open source licenses enable the "outsourcing" of functionality to other projects. The resulting shared use of code across projects may be related to the total level of reuse and the cost of innovation. Future research needs to identify the factors that drive such outsourcing behavior.

As table 2 shows, certain components are reused more frequently than others. Future research should analyze the characteristics of reused components in order to predict the frequency of component reuse. This study should inform the design for reuse. The results from the present study pertaining to the search and maintenance costs, and the trust in components should inform

hypotheses about component characteristics that lead to high reuse frequencies. Future advancement of the research on the above research issues should be built on a categorization of components in terms of functionality and, possibly, quality in order to theorize about reuse success.

Finally, we found that long-term and more active developers performed more code reuse in absolute, but not in relative terms (see footnote 12). The controls showed that the relative pattern of reuse activity over the total time of coding activity remained unchanged across developers who had coded for the project for more or less than 50% of the observation period preceding the reuse incident. Future research needs to investigate if the cause of this is individual developer learning, familiarity with the code base, stronger specialization in the functionality of the software, or other reasons.

At least two limitations apply to this research. First, the study focused on code reuse, specifically component reuse. Data on single accredited lines of code was based on interviews and on an automatic analysis. By definition, the reuse of these lines of code requires developers to credit those who wrote these lines in the code modification comment made in their own projects. Considering the sheer amount of existing open source projects, non-credited lines of code reuse is close to impossible to identify. Therefore, the correctness of the accredited lines of code-analysis cannot be guaranteed. Yet, the developers stated that giving credit for reused code is a social norm when sharing code across projects (see also Fauchart and von Hippel, 2006). Future research into code reuse might need to capture accredited lines of code reuse through survey data in addition to components.

Second, the initial sampling only included projects in active development. Since active development enables the observation of code reuse practice in real time, the analysis could combine interview material with current examples and easy-to-verify component origins (e.g., whether the component was under active development in another project). Defunct projects were excluded. In order to understand the full performance implications of code reuse, future research should compare code reuse in successful and failed projects.

Implications for management practice

Open source software projects offer a vast repository of readily usable software for almost all purposes. Within the limits of the licenses and the mechanisms that communities apply to protect their work (O'Mahony, 2003), this software can be used, reused, and built upon freely (see also Henkel, 2006). This corresponds to the advantages of black-box reuse in component markets (Ravichendran and Rothenberger, 2003) with the difference that the open source components are available for free. Commercial software developers may observe and learn from open source software developers' work. The available repository of knowledge and code could lower the probability of reinventing the wheel for firms and communities by offering methods and algorithms from open source solutions for reuse. Thus, managers of software firms should encourage and support the learning process for developers who spend time looking at available open source software.

Managers who allocate developer resources to open source software projects will possibly see new practices of innovation develop in their firms. Developers exposed to open source software development might bring practices into the firm that favor knowledge reuse over the reinvention of the wheel and introduce elements of an open source software development culture that could change the firm's internal culture.

An organization-wide, corporate reuse program is not a prerequisite for code reuse. This is an important lesson for management practice. Information about the popularity of software may substitute a costly certification process and enhance the developer's trust in the code. This study also showed that strong incentives for code reuse exist if the software developers act as "software entrepreneurs". Software developers who are compensated for task achievement, rather than time spent, have incentives to cut development costs and to reuse existing functionality. This could imply that if a software firm creates an entrepreneurial organization for its software developers, it may complement the importance of other reward-based incentives.

The equivalent to incentives in corporate reuse programs in the context of open source software development could help managers structure more effective non-monetary incentives. Recognition and extrinsic awards were found to promote code reuse in firms (Poulin, 1995; Isoda, 1995). In open source software, the continuous maintenance of reused components by others created the perception of free maintenance for the reusing developers. Avoiding to write a component

from scratch combined with the free (external) maintenance provides incentive for reuse. Possibly, managers allowing developers to self-assign tasks may achieve the necessary level of component maintenance to encourage reuse and can further facilitate reuse by separating the maintenance and reuse of existing components.

When inspecting the component reuse inventory, established and well-known low-level components can be identified, such as encryption software (OpenSSL), compression software (zlib), databases (MySQL), or graphical toolkits (GTK). These have all proven useful to a large audience of users and developers. The reuse behavior of open source software developers also informs the potential commercial suppliers of software components about the structure of this "emerging market". Similarly, experience from corporate component reuse shows that domain-independent reuse (often low-level components) is easier than domain-specific reuse due to lower adaptation costs (Poulin, 1995; Ravichandran and Rothenberger, 2003).

Interviews revealed that open source software developers work under severe time and skill constraints. The self-inflicted pressure to release a working product leads to efficiency thinking and economic behavior with regards to the utilization of scarce resources. The insights from innovation process research in open source software continue more than ever to be useful to researchers studying innovation in firms (particularly software firms), since similar economics of innovation apply in both contexts. The limitations of researching into the alleged "hobbyist culture" of open source software (Carbon et al., 2001; Moody, 2001) does not apply when studying the social and technical processes of innovation in open source software development. As shown, there are important lessons for researchers and managers considering innovation in both contexts. Work on open source software development and its commercial counterpart will mutually benefit from an exchange of results and jointly they can contribute to an extended theory and insights into private-collective innovation.

OPENING UP DESIGN SCIENCE:

THE CHALLENGE OF DESIGNING FOR REUSE AND JOINT DEVELOPMENT

Georg von Krogh
Stefan Haefliger

Chair of Strategic Management and Innovation
Department of Management, Technology, and Economics (MTEC)
ETH Zürich, Switzerland

Published in the Journal of Strategic Information Systems, vol 19 (2010) pages 232-241

The purpose of this paper is to advance design science by developing a framework for research on reuse and the relationship between external IT artifacts and their users. A design science approach to IS research needs to grapple with the fact that a number of relevant, economically attractive, external IT artifacts cannot be designed from scratch nor meaningfully evaluated based on the current state of development, and so design science research will struggle with incomplete cycles of design, relevance, and rigor. We suggest a strategic research agenda that integrates the design of the relationship between an external IT artifact and the user by considering the impact artifacts exert on users. Three dimensions derived from adaptive structuration theory inform our framework on three levels of design granularity (middle management, top management, and entrepreneur): agenda considers the dynamic properties of technological objects, adaptability refers to the functional affordance of external artifacts in development, and auspice captures the symbolic expression and scope for interpretation. We derive implications for research design.

INTRODUCTION

Information systems (IS) pervade everyday organizational life. Managers and IS professionals build and evaluate IT artifacts, such as vocabulary, symbols, models, algorithms, procedures, and instantiations, tailored to organizational needs in order to solve problems that, until now, could not be addressed by information technology. The design science approach in IS established rigorous research guidelines that foster contributions to the problem-oriented, innovative, and effective creation, deployment, and evaluation of IT artifacts in organizations (Hevner et al., 2004; March and Storey, 2008; March and Smith, 1995). However, a growing number of IT artifacts are not only created outside the organization, they also extend beyond the organization in terms of both complexity and dynamics (Elbanna, 2010). As a result, any one designer's ability to fully understand and influence overall development remains limited. Working with systems and environments such as GNU Linux, Apache, or Mozilla, to name just a few of the largest and most popular open source (OS) families of programs, managers and designers face the challenge of using external IT artifacts—existing artifacts developed outside their organization. This design activity that encompasses relating to external IT artifacts is only partially understood: as reuse across organizations (Ravichandran and Rothenberger, 2003; Haefliger et al., 2008) and as community relations entertained by firms (Shah, 2006; Dahlander, 2007).

The emerging literature on reuse of external IT artifacts considers search and adaptation efforts (Bonaccorsi et al., 2006; Majchrzak et al., 2006; West, 2003), whereas the literature on community relations emphasizes evaluation and sharing of IT artifacts (Henkel, 2006; Dahlander and Wallin, 2007; Dahlander, 2007; Stuermer et al., 2009). These design activities, while effective in tackling the challenge of dealing with external artifacts, partially ignore the systematic context difference between the external developers and the designers and users within the adopting organization. We currently lack a comprehensive framework that could inform design science research on the use of externally developed IT artifacts, in particular adaptation and evaluation. Crucially, use occurs in a different context from development and designers must be made aware of the effects external IT artifacts can have on use within the organization (Ciborra, 1998). Starting with the search for an IT artifact and problem formulation all the way through the adoption, development and internal evaluation, understanding the effects of use and context, that may limit "degrees of freedom" in design, has become a priority for IS researchers and practitioners. A strategic perspective reinforces the urgency because successful information systems (e.g. for knowledge management) rely on accessible and well integrated IT artifacts (Butler et al., 2008; Massey et al., 2002), and integration refers to the everyday context of use in an organization.

The purpose of this paper is to advance design science by developing a framework for research on reuse and the relationship between external IT artifacts and their users. We seek to advance avenues for future research on IS through the design science approach by formulating and grounding a set of research questions. In the next section, we briefly introduce the literature on the design of IT artifacts, and show the importance of a research thrust on designing and relating to externally designed IT artifacts. Next, we develop a framework of research questions to guide future work in this particular area. Before concluding, we discuss the implications of our framework for research design and for the role and focus of the design science researcher.

DESIGN AND RELATE: AN OVERVIEW OF THE LITERATURE

Design science was originally conceived within engineering and computer science and aimed at problem solving in these areas (Simon, 1996). Today, design science is pervasive in several academic disciplines that build artifacts, such as mechanical or medical engineering, biotechnology, construction engineering, and architecture. In IS, design science evolved into a coherent body of theory and research on design and action (Gregor, 2006). It opened vast opportunities for predicting and observing the interaction between researchers, designers, users, organizations, and the evolving artifact (Cross, 2007; Markus, et al. 2002; Hevner et al., 2004; Banker and Kauffman, 2004; Gregor, 2006). Hevner and colleagues (2004) developed a foundational design science approach to IS research consisting of two activities, the initial development of artifacts and their subsequent justification and evaluation. They based their study on business needs originating with people, organizations, and technology, as well as theoretical foundations and research methods. More specifically, Hevner (2007) posited three cycles in design science: the relevance cycle that connects design science research and the problem environment through the specification of requirements and field testing; the design cycle that connects building and evaluating artifacts; and the rigor cycle that connects design science research and developing knowledge bases. Hevner et al. (2004) distilled the practical aspects of design science into seven pivotal IS guidelines: 1) create an artifact that addresses an organizational problem; 2) ensure the problem is relevant to business; 3) evaluate the utility of the design in view of the needs or problems it is created to address; 4) contribute to academic and practical knowledge through the new artifact, methods, or foundations; 5) use rigorous methods when creating and evaluating the artifact; 6) search for an effective artifact using available means to reach a desired end, within the (legal) constrains set by the problem environment; 7) communicate design outcomes to managers and academics.

As these guidelines show, one undisputed advantage of the design science approach is the intertwined nature of artifact design and the process of researching it. The distinctions between "research and design" or "observing and doing" become increasingly blurred, to the potential benefit of practice and academia alike. Design science helps researchers and managers engage in constructive dialogues: researchers to identify the most relevant and pressing research problems, and the academic IS discipline to contribute to practically useful knowledge, novel theories, and tested methodologies (Hevner, 2007). Fundamentally, design science frees the IS field of excessive technological determinism, or the simplistic view that technology is determined by rules or laws beyond human control (Hickman, 1998). It also helps IS researchers to add "truth value" to artifacts and recommendations by specifying their effectiveness and efficiency in specific situations (Iivari, 2007: 46-47). An important assumption for design science to work, however, is that context along the dimensions of people, organization, and technology is known, potentially understood or, to some limited extent, controllable by the researcher, much like an attempt to identify and unilaterally control a complex set of variables in quasi-experiments. Inside firms, the design science approach to IS research still very much relies on a notion of a cyclical process that starts with problem formulation and ends with successful implementation (Hevner, 2007; March and Storey, 2008). As Hevner suggested (2007: 89), "Good design science research often begins by identifying and representing opportunities and problems in an actual application environment." Hevner then proceeds to clarify how the application context not only provides requirements for research, but also specifies acceptance criteria for the final evaluation of the research outcome. Under such conditions, design science is rational, rigorous, and useful. Yet, with the advent of external IT artifacts—where collaborative development across organizational boundaries engages widely distributed populations of designers and users and integrates a large variety of technologies—new forms and contingencies raise an important challenge to conventional design science in IS: the context defined along multiple dimensions becomes increasingly dynamic and problematic to identify, understand, and control unilaterally. Garud, Jain, and Tuertscher (2008) alluded to this challenge when elaborating on the design approach taken by developers of collaborative efforts such as Wikipedia and GNU Linux. What the authors call "designing for incompleteness" is a cycle of evolving artifact designs that opens up questions and options for re-design.

Today, there is a growing environment populated with large and complex IT artifacts that have been in development over many years (the GNU Linux operating system, for example, began development in 1991) involving communities of thousands of individuals and organizations worldwide (von Hippel and von Krogh, 2003). The artifacts are often instantiations, (e.g. OS software licensed code), but may equally well include a vast repertoire of alternative algorithms, symbols, design methods, vocabulary, or procedures. In the light of this phenomenon, we propose two important issues to be considered for design science: reuse and community relations. First, the global search space for OS-related artifacts is virtually unconstrained. Although software repositories such as Freshmeat or SourceForge (which list more than 120,000 projects and 1.2 million developers producing a host of IT artifacts) aid firms and users conduct searches for effective artifacts, the information problem is overwhelming: depending on the constraints set by the problem environment, one does not know if an even more effective artifact could be available "out there," or if some individual or firm is working on solving the exact same problem at this very moment. Haefliger et al. (2008) found that OS software developers sometimes use software repositories for searches, but that they rely more strongly on their contacts with other trusted developers when identifying effective and efficient artifacts. It should also be noted that the reuse literature in IS focuses on cost savings through internal reuse programs (Frakes and Isoda, 1994; Kim and Stohr, 1998; Cybulski et al., 1998; Lynex and Layzell, 1998) without formulating methods to decide which external IT artifacts to reuse in order to design new or better information systems.

Second, current and future individuals and organizations involved in the design of the external IT artifact may be unknown to the firm, its designers, and users, as will their problems and business needs. Yet, an increasing number of managers and designers in firms see great promise in OS-related IT artifacts developed by global communities, and have started to adapt these to internal needs, contribute to their collective development, and build relationships with the communities (Bonaccorsi et al., 2006; Henkel, 2006; Dahlander and Wallin, 2006). Additionally, a design science approach to IS research needs to answer the question of how best to relate to the community to jointly build new and effective IT artifacts. OS software and related artifacts often evolve without a clear "roadmap," and the boundaries of their future functionality may be highly uncertain. For example, in very large systems, such as the Debian GNU Linux Distribution, continuously changing groups of committers with privileged access to the software development add or remove software components at a high pace. Their decisions to remove or add components are often based on the intensity of certain components' use in the community (Maillart et al., 2008). Any single organization is but one (possibly important) contributor to the community's project. Given the importance of the nature of each contribution to the collective design agenda, design science needs methods to determine what to contribute for joint development.

One response to these and other challenges would be to limit the application of design science to the stable and well-defined organizational context where it first originated, and then create new theories and research approaches for these novel and evolving phenomena. We believe this would be the wrong response, as it would fail to realize the strong potential that design science in IS holds for the phenomenon of OS software. Rather, we argue that designing the relationship between the artifact and the user is a vital, if not central, aspect of design when external IT artifacts are developed by global communities. This calls for a new research framework with a set of questions to be tackled by design science researchers. We turn to this point next.

TOWARD A NEW RESEARCH FRAMEWORK FOR DESIGN SCIENCE

A design science approach to IS research needs to grapple with the fact that a number of relevant external IT artifacts cannot be designed from scratch nor meaningfully evaluated based on the current state of development, and so design science research will struggle with incomplete cycles of design, relevance, and rigor. Three observations lead to a new research agenda that integrates the design of the relationship between an external IT artifact and the user.

1. Economically attractive, complex, and large IT artifacts are publicly available and can be used for free.
2. Their complexity may prevent members of one organization from fully knowing or understanding the context of the artifact, including the dimensions of people, organization, and technology involved in its design.
3. The artifact is being developed by a global community, which may or may not be open to artifact modifications and suggestions for improvement as part of the design process. The sustainable development and maintenance of the artifact remain outside the full control of any one organization.

These observations do not imply that contributors to collective innovation processes are not open to influence, but that joining or even influencing the development agenda can be difficult to achieve in a manner satisfactory to the user (O'Mahony and Ferraro, 2007; Spaeth et al., 2008). The first observation justifies economic interest in external IT artifacts. The second points to the challenge of reuse because, with only a partial understanding of context, reuse decisions become complex design matters. The third observation draws attention to community organization (Sawhney and Prandelli, 2001; Lee and Cole, 2003) and, thus, the importance of community relations by anyone wishing to invest in, use, and control external IT artifacts.

Reuse and community relations are positions on two dimensions that need to be taken into consideration when designing the relationship between an external artifact and the user. Reuse can mean adopt-as-is or adapt-to-fit, a choice with respective trade-offs. A purely reactive approach to reuse (adopt-as-is) may lead to unsatisfactory results in terms of fit with organizational needs. A focus on adaptation (adapt-to-fit), on the other hand, may miss the advantages of external maintenance and future improvements. The approach to community relations implies similar trade-offs: founding and nurturing a community is costly but rewarding in terms of influence over the artifact development agenda (Spaeth et al., 2010), whereas remote participation may yield no influence at all. Community relations can mean foundation or sponsorship by a firm—such as IBM's decision to release the Eclipse software development platform under an OS license and help build a foundation to manage it—or remote participation, evidenced in the roughly 170 member organizations contributing to the Eclipse platform. A statement in a recent keynote address by Dan Frye, Vice President of OS software at IBM, shows how delicate this trade-off is for Eclipse:

> *"For IBM, one of the hardest lessons it had to learn was one about control. Mainly, there is none. There is nothing that we can do to control individuals or communities, and if you try, you make things worse. What you need is influence. It goes back to the most important lesson, which is to give back to the community and develop expertise. You'll find that if your developers are working with a community, that over time they'll develop influence and that influence will allow you to get things done." (Quoted in Kerner, 2010.)*

While the trade-off may prevent firms and other users from engaging in and developing community relations, research by Joachim Henkel (2006) shows that reciprocity in community relations can offer benefits for the participant firm, such as external bug fixes and software improvements, and an enhanced technical reputation.

The trade-offs apparent in reuse and community relations are contingent upon the way designers and users in organizations understand and relate to external contexts, technologies, and community organization. Existing artifacts are known to impact users (Orlikowski, 1992; DeSanctis and Poole, 1994)—an insight that may be fruitfully combined with a design science approach that views artifacts as the creative outcome resulting

from a precise understanding of the gaps between the current and desired states of organizational information systems. We propose that a future design science approach should consider more closely the relationship between a given and adaptable artifact and the user. Adaptive structuration theory (AST) is particularly useful to help devise a framework toward this end (DeSanctis and Poole, 1994; Markus and Silver, 2008)[57]. AST builds on the work by sociologist Anthony Giddens, and originates from an effort to combine various institutional and decision making theories in prior IS research. AST examines technological and organizational change from the types of structure offered by advanced technologies, and the structures that emerge in human action as people interact with these technologies. The theory is relevant to our argument because it aims at improving the design and implementation of new technologies (DeSanctis and Poole, 1994). Correspondingly, we suggest the effects of IS on users should be considered along the three dimensions identified by Markus and Silver (2008) in their later critique and advancement of AST: technological object, functional affordances, and symbolic expression.

Acknowledging that a global community may continuously develop artifacts, the dimensions of AST take on a dynamic aspect ranging from current to future activities and states, which needs to be considered by designers. The characteristics of the technological object can be subsumed under "agenda," consisting of realized, planned, or evolving technology. The functional affordances can be termed "adaptability," since they result from an informed perception of functions by designers ready to invest considerable resources to adapt a given functionality. Third, "auspice" is the promise of symbolic expression that covers the interpretative scope of designers and users who choose to work with a "foreign" external IT artifact developed by a global community. In ancient Rome, an auspice referred to patterns as "signs from the gods," such as a flock of bird or the appetite of chickens, interpreted by an augur. Analogously, designers read patterns in the development of an evolving complex artifact as well as the many weak and strong signals emerging from the behavior of the community. Patterns in development might include software components removed from or added to the OS software repository, implementations of algorithms, choice of programming language, API specifications, waiting times for bug fixing, release histories, or evolving documentation of code. The behavior of the community may include messages posted to mailing lists, thread lengths of discussions, the developers or users who choose to participate in certain discussions, the types of mailing list used (e.g. technical topics versus general lists), frequently asked questions, helping behavior, etc. The community often cultivates and communicates specific values that might contradict organizational values and otherwise interfere with working routines, rhythms, and interaction patterns in the organization. Within this structuring process, the role of the designer can be considered at different levels of granularity: from middle management to top management, and from an industry perspective, where designers are entrepreneurs (Steyaert, 2007) using given artifacts and combining them with their own perception of markets, user needs, and technological visions. Table 1 presents a number of critical design issues that risk remaining invisible when ignoring the relationship between external IT artifacts and users.

Agenda

As designers, middle management need to anticipate the planned, realized, and evolving characteristics of the external IT artifact, to understand and evaluate its potential utility for their organizational problem, and ultimately for the business relevance of the artifact. In a global community the number of components that constitute the artifact may grow exponentially, satisfying some user's personal needs (von Hippel, 2001). In the process, some components may become increasingly peripheral, whereas others may take on an important role in the overall design, channeling the efforts of a greater part of the community toward upgrading and ensuring compatibility with other components. Nokia's development of the OS software platform Maemo is a case in point. While the platform includes OS software like GNU Linux, GTK, and Gnome, in addition to many other user de-

[57] March and Smith (1995) suggested that adaptive structuration theory is a prominent example of an approach to theorizing about the "why" and "how" effects in IS design. For example, an issue that connects the two theories is why particular artifacts (constructs, models, methods, and instantiations) work. It should be noted, however, that Markus et al. (2002) elected to talk about "design theory" when describing knowledge management systems linking theory and instantiations, rather than design science, which focuses more on design as an activity. For more on this, see Gregor (2006).

to many other user developed instantiations (e.g. mapping software or star gazer), Nokia has chosen to keep some software proprietary, including user experience-related software components. However, for Nokia, the design of an optimal user experience needs to be kept consistent with

	Design granularity		
	Middle management (group level)	Top management (firm level)	Entrepreneur (industry level)
Agenda (dynamic properties of technological objects)	Design the adoption: How can global community dynamics be understood to anticipate relevant technologcial changes?	Design a reuse strategy: How do external artifacts complement the firm's sourcing strategy?	Design new business models: How do new technological developments give rise to new markets?
Adaptability (functional affordance of external artifact in development)	Design the contribution: How can a match between the external artifact and internal user needs be achieved? How to reliably evaluate security, reliability, sustainability?	Design community relations: How can future internal needs be matched with current functions identified?	Design for market demand: How can functional needs be matched with market demand?
Auspice (symbolic expression and scope for interpretation)	Design for goodwill: How can the usefullness be communicated and explored given diverse internal needs? How to evaluate and deal with incoherences in the system?	Design for openness: How can we overcome the NIH syndrome and, e.g. promote acceptance of FLOSS, or diversity in IS? How to foster creativity?	Design an environment for collaboration: How can community work translate to entrepreneurial activity and lead to spin-offs?

Table 13: Understanding the relationship between the external IT artifact and the user: a research framework for advancing design science in IS

the functionality of these evolving artifacts. Thus, Nokia's designers are required to interact carefully with, learn about, and monitor the realized, planned, and evolving technology development and other activities of the community.

In a context where the dimensions of people, organization, and technology are in flux, the organization's overall strategic direction is an important consideration for researchers who do IS design science. A changing context can alter organizational problems, undermine their business relevance, and diminish the utility of a design. The development agenda also includes top management designing an overall reuse strategy aimed at externally developed IT artifacts, and asks how the realized, planned, and evolving artifact complements other internal strategic assets (artifacts) and their sourcing. The reuse strategy defines a scope for design and action throughout the organization, and provides direction for middle-level managers and other designers and users to identify relevant communities and technologies. Moreover, the reuse strategy also defines how externally and internally developed artifacts relate, including make-or-buy decisions at the level of artifacts, such as software components and other instantiations. Ravichandran and Rothenberger (2003) discuss this for component markets, where the difference is that components are commercial and protected by copyright.

At the industry level, the development agenda may inspire the design of new business models. A study of new entrants in the Italian software industry (Bonaccorsi et al., 2006) finds that entrepreneurship emerges on the fringes of OS software development, giving rise to new markets for products and services. Here, users design business models that involve community support, packaging, combining, selling artifacts, or providing services to other OS software users (see, for example, Fitzgerald 2006). Interestingly, the fact that artifacts rapidly evolve is in itself an entrepreneurial opportunity. Examples of entrepreneurial firms that developed new business models based on OS software include Red Hat and Suse. In a world where Linux and related software may contain numerous releases, Red Hat adds value

to the user of OS software by securing coherent and updated versioning of the software (Red Hat Linux) in combination with other artifacts. To summarize the discussion this far, the overarching research question regarding the effect of the agenda is: how can the link between existing technology and organizational needs be designed so that the use of the artifact can sustain, evolve, and grow in conjunction with the external development?

Adaptability

Adaptability requires an understanding of the potential match between the user's requirements and the functional affordances, defined as "the possibilities for goal-oriented action afforded to specified user groups by technical objects" (Markus and Silver, 2008: 622). From a dynamic perspective, middle managers are challenged to find a match and design an adaptation strategy that takes into account the specific needs of the organization regarding scale, security, reliability, stability, and more. Here, make-or-buy decisions as part of the adaptation strategy become an important object of research. What are the costs and benefits involved in adapting an external IT artifact, and at what point do net benefits of adaptation offset the cost of developing a targeted artifact inside the firm?

For top management, these dynamics imply the choice of a community of external developers or the foundation and nurturing of sub-communities to design sustainable collaboration with competitors and volunteers (Spaeth et al., 2010). Top management involvement is essential here because engagement with the community to safeguard functional affordances represents potential risks including loss of reputation, loss of trade secrets, spillover of technically sensitive information, the use of internal labor for external IT artifact development of no relevance to organizational problems, unintended or intended breach of commercial and OS software licenses, etc. While prior work has suggested investigating the roles and activities of top management by design science methods (Van Aken, 2004), little is known about their involvement in design science and how the reuse of external IT artifacts opens a new research space.

An entrepreneur's challenge is to identify and build the match between the functional affordances of the existing artifact and a new segment of users willing to pay for a specific service or complementary product. The entrepreneur holds or creates knowledge about the evolving artifacts, their goal orientation, and the existing and potential needs of users. Through involvement in the design, the entrepreneur learns about the affordances of the IT artifact and starts to relate it to new users (Steyaert, 2007). This process may result in new matches between functionality and existing demand, but it may also, through a succession of commitments, result in the creation of entirely new markets (Sarasvathy and Dew, 2005). As Sarasvathy and Dew (2005: 559) put it: "Entrepreneurs do not 'leave it' to differences in tastes or behavior to build markets. They work very hard to make tastes cohere and to concurrently embody them into particular transformations in real artifacts." The overarching research question regarding the adaptability is: how can the functional affordances of the artifact be identified and (or) created in order to meet the needs of users when both technology and requirements continually evolve?

Auspice

The symbolic expression of an IT artifact corresponds to a dynamic scope for interpretation. Auspice is a promise in need of design. Management is challenged to communicate the usefulness of a need and convey a sense of motivation among users that triggers their creativity. For example, Hevner (2007) highlights the importance of creativity and flow in problem solving for a successful design. Yet, the use of an evolving, external IT artifact is fraught with difficulties: it is "not invented here," it may come with bugs, it may be poorly documented, it does not perfectly fit internal routines and practices, it may be abandoned by the original developers, and its inner workings may never be fully understood by users and designers in the organization. Appropriation by internal designers and users, their acceptance and enthusiasm for integrating the artifact with the information architecture of the organization, can be designed, and our framework specifies the issues involved. Middle management designs for goodwill toward a specific external IT artifact. The successful use of IT artifacts depends on organizational changes in routines and business processes (Brynjolfsson and Hitt, 1996), which include complex relationships between management levels and organizational units (Mata et al., 1995). Auspice describes the relationship between the external artifact and the

user, the scope for interpretation and, hence, communication and flexibility in promoting the artifact's advantages. At the middle-management level, auspice also involves transforming internal processes to accommodate for and actively use the external IT artifact. These middle management tasks can be daunting, as any external IT artifact may be viewed with skepticism and mistrust, based on justified criticism.

For top management, designing openness becomes a strategic issue: why should the firm adopt external IT artifacts at all? Previous research has shown that IT assimilation crucially depends on top management (Armstrong and Sambamurthy, 1999) prioritizing and carrying out specific organizational initiatives. Top management may indeed have significant influence over whether external IT artifacts are generally positively received, and future research needs to substantiate the most effective strategies. In the case of Nokia's Maemo platform, significant communication was needed by top management on the direction of the company's collaboration with the community of volunteer developers contributing a variety of artifacts. A good starting point for research could be some of the world's largest technology companies, which frequently interact with and contribute to OS software communities. For example, companies such as Red Hat, IBM, Novell, Intel, and Oracle are top contributors to the Linux kernel: in 2009, there were 240 companies contributing overall[58]. The fact that some of these companies have been working with OS communities for many years seems to indicate that they value community relations highly and that they have found ways to communicate the value of openness to internal users and designers.

From an entrepreneurial perspective, understanding the symbolic expression of an artifact could translate into designing an environment for collaboration and overcoming the little understood hurdle of motivating distributed individuals to share or contribute to a new business. While it is clear that many entrepreneurial firms have emerged on the fringes of the OS software phenomenon as discussed above (we also see software vendors being formed around the Maemo platform), a large research gap exists concerning the community's engagement in the formation of these business. The emergence of entrepreneurs may indeed create an imbalance in the incentive structure that safeguards OS development. Some voluntary users and designers who do not expect to benefit economically, at the same levels as entrepreneurs, may defect from the development effort, which could lead to an undersupply of the external IT artifact (see discussion by Young and Rohm, 1999, on establishing Red Hat and the sensitivity to voluntary Linux developers). The overarching question regarding the auspice is: how can the promise of using an external IT artifact be communicated and promoted in order to enable efficiency and foster creativity?

Table 1 summarizes a host of design tasks that are relevant, but neither obvious nor easily tackled by existing approaches to design science in IS. A focus on the relationship between artifact and users can broaden the scope of the design science approach considerably and enrich research by incorporating theory from other fields, such as structuration theory, motivation theory (flow, cognitive dissonance, self-determination), private-collective innovation theory, or behavioral economics (reciprocity, fairness, altruism). If the technology is developed outside the influence of any one organization, and if both technical characteristics and internal needs continue to evolve, the great promise of a design approach lies in the carefully structured, rigorous, and repeated questioning, understanding, and designing of the relationship between the artifact and the user.

[58] For recent statistics see lwn.net or for a comprehensive update for 2009 see Kroah-Hartman et al. (2009).

IMPLICATIONS FOR RESEARCH DESIGNS

In the following, we discuss the implications of our framework for design science research and the researcher to indicate potential avenues for future work. The result of design science research is the production of an artifact such as a construct, model, method, or instantiation. Thus, an important strength of the design science approach in IS is its close link to action and practice (March and Smith, 1995; Hevner et al., 2004; Hevner, 2007; Sein et al., 2010). Similarly, in the framework we propose, the result is the artifact that emerges from a collaboration where the design approach includes not only the artifact itself, but also the relationship between designers and users (spread out across several communities or firms). This broader focus implies extensions to the design science research process in IS.

Recall that design science consists of cycles of activities to foster academic and practical knowledge; the design cycle (build and evaluate), the relevance cycle (specify requirements, field tests), and the rigor cycle (grounding, additions to knowledge base). The framework developed in the last section raises the question of the location of these cycles and the role of the researcher within them. If the problem environment for one artifact consists of multiple and changing people, organizations, and technologies, the focus of design science and the position of the researcher become ambiguous. We cannot fix a general position but need to consider a space with multiple dimensions and multiple perspectives from which an artifact is external. Design science, then, can be understood along a vector in a three-dimensional space where its direction captures the focus of research (the user as researcher, the community, and the firm) and its scope captures the extent to which research cycles include the relationship between external artifact and user. We use the term "vector" because individual researchers can choose to focus to a greater or lesser degree on one or more players and their relationships in the overall research context.

First, let us consider direction. The orientation of design science research can be towards the researcher as user. An important motivation for OS software is for users to solve their own problems, through the building and evaluation of an artifact (von Hippel, 2001). Here, the artifact can be built from scratch or reused from the wide offerings in an OS environment. A prerequisite, however, is for researchers to build and evaluate an artifact with future potential reuse in mind. In many engineering fields, researchers build IT artifacts to solve critical scientific problems, and release them with the publications of their research results. In geography, for example, researchers have built numerous OS software components for geospatial applications. The purpose of these systems ranges from helping researchers to do more accurate mapping and tracking of geophysical changes, to better analysis through geometric algorithms (Steiniger and Bocher, 2009). So, too, in the field of economics, where authors have argued for the need for the discipline to build and evaluate new OS IT artifacts such as software packages for more effective statistic analysis (Yalta and Yalta, 2010). When researchers share these artifacts in the way the examples show, it may help advance a discipline more efficiently and effectively.

Design science research can be oriented toward a community that engages in collective building and evaluation of reusable artifacts. Researching OS software communities is nothing new, of course, but following the principal cycles of design science (Hevner, 2007) researchers are more than passive observers—they must join the community, build and evaluate through participating in discussion forums, testing software, reporting bugs, fixing bugs, documenting, writing code, coordinating projects, and so on. The focus of design science research would be on the extent to which reusable artifacts that fit with the emerging needs and problems of the community can be built. To our knowledge, this type of research design is currently almost non-existent (an exception is Bodker et al., 2007), but we think it will be instrumental for the development of our academic and practical understanding of OS communities. For example, community oriented design science will allow us to examine in detail a number of context-sensitive issues linked to the building and evaluation of an artifact. For example: if, why, and how do small changes to an artifact initiate supportive or negative reactions from the community? When and how does the community perform search? What is the form or content of discussions closed to people outside the small circle of OS developers? What changes in an artifact relate to or emerge from the ongoing discussion on project developer lists?

Finally, design science research can be oriented toward the firm as the user organization. Here

researchers place themselves within the organization, jointly searching for and evaluating communities and external IT artifacts for reuse. Design science has contributed significantly to information systems research in this domain; their contributions have strategic implications for how to improve working environments in organizations and leverage information systems for business purposes. Consequently, it is here where our framework most obviously builds on prior work and extends the design science approach to include the relationship to external IT artifacts. We are not aware of scholars using the three design science research cycles in conducting research that takes into account the impact of the dynamic properties of external IT artifacts under development, of their functional affordance, and of their symbolic expression. However, Hevner (2007) sees parallels between action research (e.g. Susman and Evered, 1978) and design science research, and suggests that both could benefit from each other. Very recently, Sein and colleagues (2011) developed a methodology that combines action research and design science research by building on their own action research and on prior work in design science that explicitly takes into account the organizational context of the design process (Gregor and Jones, 2007; Orlikowski and Iacono, 2001; Orlikowski and Scott, 2008). There are several studies in IS that report having used an action research methodology concerning the implementation of OS software in public organizations, developing economies, and the educational system. For example, Braa and Hedberg (2002) report on an action research program to develop a health information system based on OS software for African countries, including a number of challenges such as acceptance of the new systems by local users. However, many of these studies fall prey to a common criticism that action research tends to favor strong relevance over academic rigor. An exception here is Fitzgerald and Kenny (2004) who provide a rigorous account of OS software implementation in a large Irish hospital. Their study identifies challenges such as the shift of mindset needed to work with OS software, as well as the resistance among staff who felt their expertise was threatened as the hospital abandoned popular commercial software. Yet, while the study pays significant attention to the change resulting from the OS software implementation, reporting case-based findings of high relevance to the IS discipline, it is not a direct application of a design science research approach. In particular, it does not place the researcher in the focal position of external IT artifact design and reuse. This will need to be the orientation of future design science research.

Second, the scope of the vector represents the building and evaluation of the artifact itself, dealing with the extent to which researchers take into account relationships with external IT artifacts. It is here that our framework suggests specific questions for design science research, specifying methods to decide on the fundamental trade-offs in reuse as well as community relations. While the application of our framework is not limited to firms or communities, the questions are most fruitfully investigated in a context where more than one individual (the researcher) is affected by external IT artifacts and organizational IS challenges can be approached with the relationship explicitly considered.

As an iterative process, the cycles imply that not only the external artifact underlies a build and evaluation process but also the nature of the relationship with the community and the design efforts that can be controlled by the internal designers. The search for a reuse target is a process that compares expected search costs with estimated costs of designing in-house: only if the efforts to design an artifact by internal designers are considered to be higher than the potential identification of a reusable artifact will search be undertaken (Haefliger et al., 2008). All three dimensions of our framework should be considered when making decisions about reuse and community relations. First, regarding agenda, should search focus on currently available artifacts or include, or even focus on, technological roadmaps and development agendas set by communities? Any candidate artifact needs to be considered in terms of its relationship with internal designers and users. This means that the estimated adoption effort depends on current and likely future needs. The design of a search process is already known to be fraught with "uncontrollable forces in the environment" (Hevner et al., 2004: 88), without taking into consideration the relationship between an externally available artifact and internal designers and users at a given time. The resulting complexity may well overtax predefined search procedures and call for new search heuristics.

An externally developed system comes with a history. Its architecture has evolved based on the needs of the community developing it and its

features may make it a promising candidate for reuse. The community's agenda to develop the system further may be in line with internal needs and first tests might suggest adoption. Designing for adoption requires the design science researcher to answer a number of specific questions such as: how does the system technology interact with the internal technical environment? Do internal designers sufficiently understand the system to make use of it? Is compatibility ensured across key processes? What is the release history of the community and can we expect the community to continue to maintain and release updated versions? Is the community responsive, offering documentation or a help forum?

Second, regarding adaptability, searching for the right contribution to a community, given the functional affordance of the external artifact and the pre-defined and emerging needs of internal designers, may result in experimentation and improvisation with new approaches such as design for incompleteness (Garud et al., 2008). Designing the contribution means evaluating the gap between existing (and potential) functionality and internal needs in terms of adapting the system. This may entail contributing to the community in order to benefit from the community's future improvements (Dahlander and Wallin, 2006). Design evaluation, here, means gauging the potential community relations that allow quality improvements to flow both ways. Is there mutual understanding in terms of quality, usability, and security standards? If not, can new requirements be introduced into the community by contributing and relating to the community on productive and friendly terms?

Third, the extended design cycles may benefit particularly from the auspice that external artifacts carry. Wide-spread use and popularity of an artifact designed by a global community may signal quality and the visibility of the artifact may reduce the search cost (e.g. components in the GNU Linux distribution Debian)—but such artifacts may also have an "image" that could deter internal users from embracing them. Consider the ease of identifying software distributed under the Apache umbrella of projects hosted by the Apache Software Foundation. The public fame of Apache and the outstanding reputation of the Apache web server enhances the visibility of affiliated programs, such as Lenya, an OS content management system. Opting for a highly visible artifact may increase internal acceptance and a perception of openness in the firm. However, it may also have a reverse effect and aggravate a "not invented here" response by internal designers, or raise flags about potential legal issues when using free and OS software rather than a commercial alternative. Designing for goodwill has to do with internal acceptance and leverage of the community's work despite uncertain community relations. How is the community perceived by users and internal designers? Is close interaction with the community desirable and promising? Does the community's work and practice (including visible signs, communication style, quality perception, etc.) inspire the creativity of internal designers and users?

Establishing research cycles in design science that take into account the dimensions outlined in our framework shows the complexity of an iterative and often intractable process. The AST approach suggested in our framework emphasizes the mutual influence between the reuse of artifacts and community relation decisions and the ability of internal designers to build and evaluate effective IT artifacts in collaboration with external communities. Thus, the extensions to design science research contained in this framework directly inform reuse and community relation decisions, possibly to a finer level of detail and practicability than the innovation literature has so far achieved.

Conclusion

Design science has evolved into a powerful set of theories and methods of design and action in IS. A conventional approach to design science in IS benefits from a stable context along the dimensions of people, organization, and technology. We argue that the phenomenon of OS software introduces major changes in this context, calling for a new and important strategic research agenda for design science. A focus on the relationship between external artifacts and organizational users and designers can considerably broaden the scope of the design science approach and enrich research by incorporating theory from other fields, such as adaptive structuration theory (AST). Based on AST, we developed a new research framework covering agenda, adaptability, and auspice at three levels of granularity—middle management, top management, and entrepreneur. The framework sorts design tasks and challenges pertaining to the different dimensions and shows how technological artifacts impact internal designers and users. Future work needs to develop methods and processes that ultimately help designers make key decisions regarding reuse and community relations. We outline a design science research vector along which researchers can organize their research activities.

A focus on the user-artifact relationship also opens up new research topics in other areas, such as motivation theory (flow, cognitive dissonance, self-determination), private-collective innovation theory, or behavioral economics (reciprocity, fairness, altruism). If the external IT artifact is developed outside the influence of any one organization, and if both technical characteristics and internal needs continue to evolve, the great promise of a design science approach lies in the carefully structured, rigorous, and repeated questioning, understanding, and construction of the relationship between artifact and user. Working on the research areas defined by the framework may even bring forth the contours of an "open design science."

SOCIOMATERIAL TRANSACTIONS AND THE ECONOMICS OF DISTRIBUTED SYSTEMS INTEGRATION:

THE CASE OF FREE AND OPEN SOURCE SOFTWARE DEVELOPMENT

Stefan Haefliger

Chair of Strategic Management and Innovation
Department of Management, Technology, and Economics (MTEC)
ETH Zürich, Switzerland

To be submitted to the Cambridge Journal of Economics

This study explores how software developers collaborate across several projects (and organizations) to innovate and to make their software compatible with other software. Technology development is perceived as a social activity and developers manipulate the social fabric of software development for economic reasons. The developers reuse and fork Free/Libre Open Source software (FLOSS) code developed in other projects in order to build an integrated system. Referred to as sociomaterial transactions, these practices of distributed systems integration share three characteristics: transparency (openness) of technology, frequent communication, and long-term, multiple project affiliations by developers. This study contributes to the economics of systems integration and technical design by showing how developers from different organizations integrate software systems without the use of a formal, central systems integrator, and how interfaces are built collaboratively across organizations. Sociomateriality as a perspective renders the surprising empirical results amenable to an economic analysis of the integration work of software developers because it allows technology (code) to play a social role and social organization to become technically salient. Methodological and theoretical implications are discussed.

Introduction

Developers of Free/Libre Open Source software (FLOSS) organize their work in projects dedicated to specific software products (von Hippel and von Krogh, 2003). The software each project develops must function with other software products. Integrating software products requires not only time and effort but also knowledge about the technologies underpinning the various components (Brusoni and Prencipe, 2006; Brusoni, 2005). This study explores how FLOSS developers collaborate across several projects in order to integrate their software into a complex system and make it compatible with other software, without any one individual or organization taking the systems integrator role. The collaboration observed contributes evidence regarding the distributed location of systems integration activity in FLOSS projects.

FLOSS development is frequently described as a social activity (Weber, 2004; Bergquist and Ljungberg, 2001) because of the central role that FLOSS code plays in the social exchange: the exchange of code and about code lends itself naturally to a perspective of entanglement where a separation of the code from developers' intentions, claims and interpretations of the code seems artificial (Orlikowski, 2010; Orlikowski and Scott, 2008). The costly work of systems integration brings to light the role that technology plays in economic terms when it comes to bear on the developers' struggle to make code matter for their projects and get it to work in relation to a wider system of components. Projects are built on reused and forked software code and, hence, the technology is the social fabric that connects developers' work (Faulkner et al., 2010). Sharing code is part of the social activity that leads to a working, integrated system. The work organized in projects centers around the development of software, which is the artifact that helps shaping the cross-project work (Chua and Yeow, 2010). Given these entanglements, I try to analytically untangle actors and artifact in order to speak to a baseline model and connect the emerging perspective of sociomateriality to long established scholarship on technical design and the economics of systems integration.

Building and changing complex systems is believed to require a systems integrator who is able to understand and to some extent shape (a) the technological fields underpinning the variety of components, (b) the organizational and relational requirements for integrating the activities of multiple actors, and (c) the evolution of coordination (Brusoni, 2005; Miller et al., 1995; Prencipe, 2000; 2003). Among students of systems integration, the notion seems to prevail that "the more dispersed is the production of knowledge and the more complex are the products, the higher also are the requirements of explicit integrative capabilities embodied in systems integrators" (Dosi et al., 2003: 110).

The management of technical interfaces and the corresponding organizational structure is a dynamic process (Sosa et al., 2003). The model most appropriate for this analysis is loose organizational coupling (Orton and Weick, 1990; Brusoni et al., 2001) because it captures the middle ground between contractual arrangements that can create a market for technology (decoupled) on the one hand and hierarchical organization based on authority and control (tightly coupled) on the other hand. Organizational coupling thus served as a baseline model to formulate expectations for systems integration activity derived from the literature.

In order to identify patterns of collaboration and integration activity, an environment of about thirty projects were studied, with a focus on three central projects. Key developers were interviewed, and software code and communication archives (mailing lists, repositories, release notes, and documentation) analyzed. The types of collaborative activities in which developers engaged were categorized, and then analyzed in terms of authorship, affiliations, and technical impact.

Two types of transactions and three enabling conditions led to integrated systems without centralized system integration and an efficient handling of technical uncertainty and multiple functional objectives. The *transactions* include (1) the sharing of code, in 50 instances, and (2) friendly forking, in 23 instances. The *enabling conditions* include (1) transparency provided by the regular publication of code (encouraged by Free and Open Source software licenses), (2) the frequent exchange of messages and comments among developers from different projects, and (3) long-term interactions in two forms: developers who

are substantively engaged in several related projects (facilitating shared code), and authors of forks and branches who continued to be involved in the main project even as they pursue their separate trajectories. I refer to these findings as "sociomaterial transactions" across loosely coupled organizations, which describe how a complex product can be developed and integrated.

The remainder of this paper is organized as follows: Section 2 explores the theoretical background, Section 3 introduces the Free Java environment studied, which represents a network of projects that produces components of a system called a runtime environment, and describes the research design. Section 4 discusses the results, and Section 5 concludes and Sections 6 and 7 highlight the implications for theory and methods, and limitations and future research.

THEORETICAL BACKGROUND— SOCIOMATERIALITY, ORGANIZATIONAL COUPLING AND THE ROLE OF SYSTEMS INTEGRATION

FLOSS projects develop software that works in conjunction with other software and hardware. Projects that develop FLOSS are usually understood as virtual organizations of volunteers and company employees who contribute to a common goal of producing an envisioned software product using a small range of fairly common tools, such as mailing lists, Internet Relay Chat (IRC), central code repositories, and web-based content management systems (Lee and Cole, 2003). For the purpose of this study, a project is defined as an organization that develops and evolves a clearly defined software product (code). It consists of all the members (or developers) who contribute to the development of the software by either writing code, participating in the discussions, or by helping other members: this includes individuals who post comments but not those who only read and do not participate (so called lurkers). By core developers, I refer to individuals who received write access to the project's software repository. All developers in this study were also users in the sense that they utilized the code bases under development in their work or daily life.

As virtual projects become commonplace in organization theory (Sproull et al., 2007), their interactions and alliances have been overlooked. To the author's knowledge, there are no studies that systematically explore alliances and resource exchanges among virtual projects. There are studies of communities of practice building informal communities that bridge organizations in order to connect experts separated by organization boundaries (von Hippel, 1987; Rosenkopf et al., 2001). But the engineers who traded know-how in von Hippel's 1987 study did not build an integrated system, and studies on communities of practice often ignored the exchange of actual technology in favor of analyzing communication flows.

The literature on collective activity in user innovation (e.g. von Hippel, 2005) lacks a precise understanding of how users collaborate across the basic organizational unit, which is a project that develops and advances innovations such as software (von Hippel and von Krogh, 2003; Sawney and Prandelli, 2000). A body of research on communities of practice (Brown and Duguid, 1991) and technical communities (Rosenkopf et al., 2001) has identified the exchange of knowledge across organizational boundaries (across firms and outside of firms) and the effects on mutual learning and future alliance building. In contexts without face-to-face contact, news group members were found to exchange knowledge (in the form of messages) about their practice and build social relations (Faraj and Wasko, 2005). However, the knowledge exchanged in these studies rarely included the exchange of the technology itself, an omission that has been criticized by IT researchers (Orlikowski and Iacono, 2001).

It is this omission that points to the heart of an open debate about the role of technology in management and economics (Leonardi and Barley, 2010; Faulkner et al., 2010; Dosi and Grazzi, 2010). FLOSS code has been referred to as a public good due to its status as explicit knowledge that is non rivalrous in use and non-excludable by the design of the Open Source license (von Hippel and von Krogh, 2003). However, the question what technology is in FLOSS is more complex. Dosi and Grazzi (2010: 179) point out that technology can hardly be considered a public good due to the tacit knowledge required when attempting to use technology and the costs involved in building capabilities and learning. Taking this argument a step further and acknowledging that know-how, the capacity to act that knowledge confers, is a matter of action leads to a focus on practice (Nicolini, 2011; Suchman, 2007; Schatzki et al. 2001) and, even further, on the entanglement of developers with technology with "temporarily emergent sociomaterial realities (Orlikowski, 2010: 137)." This controversy about the inseparability of human and non-human entities in action is useful in the study of systems integration because organizational and technical boundaries are subject to negotiations and sources of significant costs to developers. Whether a sociomaterial perspective is helpful to understanding the economics of distributed systems integration is an empirical question. Thus, this study centers on the following research question:

Where and by whom are the tasks of systems integration performed in a collective of FLOSS projects, each dedicated to developing a certain software product?

The literature on technical designs and product development teaches organizational frameworks that can deal with dispersed learning processes, technical interdependencies, and uncertainty in both the rate of technological change as well as in the interdependencies of components that need to work together (Staudenmayer et al., 2005; Baldwin and Clark, 2000; Brusoni et al, 2001). Using the idea of organizational coupling (Brusoni et al., 2001; Weick, 1976; Orton and Weick, 1990), I propose a conceptual analogy and a baseline model that can inform this exploratory study of integration and collaboration between FLOSS projects.

Organizational coupling describes the choice of organizational arrangement used to manage the development, integration, and maintenance of technical systems that span across specialized organizations (Langlois and Robertson, 1992; Langlois, 2002). Three arrangements can be distinguished: decoupled, tightly coupled, and loosely coupled organizations.

Decoupled organizations can coordinate via the market because the technology is stable enough to allow for specialized organizations to provide components that interoperate thanks to standards, or design rules, without the need for continuous integration efforts (Sanchez and Mahoney, 1996). According to Baldwin and Clark (2000), the establishment of design rules lowers the costs of coordination between technical modules because the teams involved in developing each module comply with the rules, thus there is mutual compliance between modules. This helps to separate tasks among teams and ensure interoperability between the technical components. According to their analysis, information about the design rules leads to specialization across an industry because individual, component producing firms can rely on their industry partners (suppliers and customers) to produce compatible components. An example of decoupled organizations can be found in the personal computer industry (Baldwin and Clark, 2000; Brusoni et al., 2001). For software, component markets can emerge when proprietary software components match exact descriptions and standards that enable their trading between software development organization (Ravichandran and Rothenberger, 2003).

Tightly coupled organizations are thought to be good at handling component technologies with high levels of interdependency and changing customer needs (Teece, 1996; 2006). Technologies evolve and shift to completely new designs, uses, or platforms that may trigger large changes in the overall system, including the standards or design rules (Utterback, 1994; Clark, 1985). The consequences for the organization of an industry are contested. While specialized firms may accommodate the changes by entering alliance contracts early on, Chesbrough (2003a) predicts integrated organizations in order to coordinate technology development during a state of flux where standards are not fixed and subject to competition. Examples of tightly coupled organizations can be found in the hard disk drive industry in the late 1980s and early 1990s (Chesbrough, 2003a), in the telecommunications equipment industry (Brusoni et al., 2001), and Microsoft in software (Cusumano and Selby, 1995).

Loosely coupled organizations occupy a middle ground, with either lower uncertainty regarding component interdependencies or lower rates of change in the component technologies relative to tightly coupled organizations (Brusoni et al., 2001). They employ a range of integration and collaboration mechanisms that include in-house as well as outsourced development, and the management of linkages to network partners that require explicit attention due to unexpected contingencies (Brusoni et al., 2001; Staudenmayer et al., 2005; Kodama, 2006). "A key characteristic of loosely coupled network organizations is the presence of a systems integrator firm that outsources detailed design and manufacturing to specialized suppliers while maintaining in-house concept design and systems integration capabilities to coordinate the work (R&D, design, and manufacturing) of suppliers" (Brusoni et al., 2001: 617-618). Examples include the hard disk drive industry in the early and mid 1980s (Chesbrough, 2003a), and the automotive industry (Brusoni et al., 2001).

FLOSS projects are organizations that usually focus on developing a specific software product (von Hippel and von Krogh, 2003) and engage in protecting (O'Mahony, 2003) and regulating access to their work (von Krogh et al., 2003). If, as in the sample studied here, several projects produce the components of a larger system, that is the software components need to interoperate in order to be useful for the intended purpose, they can best be compared to loosely coupled organizations for three reasons. First, decoupled organizations require existing design rules and mature technologies, which are frequently not

given in FLOSS. Second, tightly coupled organizations require hierarchies that confer authority, such as task assignments to members, a property alien to most FLOSS projects. And third, FLOSS projects exhibit specialization and shared knowledge to various degrees, two aspects that determine the extent of loose coupling across projects (Orton and Weick, 1990).

The first point is the least obvious because software systems frequently start out with minimal standards that allow a new program to run on a specific operating system (such as GNU Linux), using a specific infrastructure (such as XML), or depending on pre-installed software. Since many new projects self-select their compatibility and requirements, the analogy to decoupled organizations frequently holds. Hence, if we study a context where design rules are incomplete or about to be established, we are likely to observe loosely coupled organizations.

Second, tightly coupled organizations are unlikely to emerge in FLOSS due to the partly voluntary character of contributions and the self-selection of tasks within projects (Lee and Cole, 2003; von Krogh et al., 2003). Hierarchies exist, however, and large FLOSS projects may use release deadlines, social norms, clearly defined roles, or charismatic leadership to convey a sense of responsibility and discipline (for the example of the Linux kernel, see Moon and Sproull, 2000). More frequently, and this is the third argument, small FLOSS projects are observed (Krishnamurthy, 2002) with varying degrees of specialization and scope that, given the need, will have to expend considerable effort to achieve integration with other programs. A recent study of code reuse in Open Source software documented the vast reuse activities by FLOSS developers and the accompanying work needed to make reuse successful (Haefliger et al., 2008). The study focused on the reuse of code from the perspective of the re-user only rather than on the production of shared code as carried out by potentially more than one party.

Using the analogy of loosely coupled organizations as a baseline model, we can formulate three predictions for the tasks of systems integration across FLOSS projects:

First, collaboration is a prerequisite for integration, and the building of design rules requires the exchange of knowledge and expertise. The integration of technical systems critically depends on the knowledge required to understand the various disciplines underlying the system components. In their thorough case study, Brusoni and Prencipe (2006) demonstrate that a modular technical system does not imply a modular organization (see also Hoetker, 2006) nor modularization in the knowledge domain. Innovation at the system level instead requires integration of knowledge across the organization (Brusoni and Prencipe, 2006) and the, potentially costly, establishment of design rules. The authors showed that aligning production phases with the corresponding information flows demanded and triggered communication between experts that had previously been separated, and previously tacit knowledge had to be made explicit. We should therefore expect that systems integration across FLOSS projects may require collaboration among previously unconnected programmers, because the integration of work flows and processes only succeeds if appropriate design rules are built first, and if knowledge overlaps can be created.

Second, systems integration is costly because it creates specific tasks that need management, central or distributed. The economics of systems integration creates possibly the biggest puzzle when formulating predictions for FLOSS development. The role of the systems integrator is central to loosely coupled organizations and, it seems, indispensable for an environment of complex technologies and dispersed knowledge production (Dosi et al., 2003; Prencipe, 2003). FLOSS projects arguably match the description of loosely coupled organizations and, thus, should incur considerable costs for making a system of integrated software components work. However, we rarely observe FLOSS projects dedicated to systems integration. This contradiction warrants a special focus on the handling of the economic aspects of systems integration, in particular the location or distribution of integration efforts.

Software developers set up or break component (module) boundaries as they see fit given the state of the technology and given the need to transfer information and code to other developers. Baldwin (2008) describes online communities, such as FLOSS projects, as enabling unencapsulated, transaction-free zones. "Transaction-free zones in which agents freely access and transfer valuable materials and information are necessary for most forms of efficient production." (Baldwin, 2008: 182) In contrast to firms, however, FLOSS developers perceive advantages in free access to their code by all (von Hippel and von Krogh, 2003) and do not encapsulate their productive system. Unencapsulated, transaction-free zones

may lower the transaction costs associated with systems integration tasks.

Third, systems integrators apply mechanisms to deal with uncertainty, both in technical change and in the interdependencies. Systems integration requires explicit attention to uncertain developments in both technological components and dependencies among the components. The management of these contingencies may include creative solutions that run counter to established wisdom, such as adding organizational structure to handle external relationships and introducing new, shared responsibilities (Staudenmayer et al., 2005), or establishing monitoring systems that also facilitate communication (Argyres, 1999). Hence, predicting creative solutions to manage uncertainty is difficult. We may expect FLOSS projects, similar to firms, to rely on a range of organizational mechanisms that lie between in-house production and outsourcing.

The baseline model leads to a research agenda that starts at the deep end of collaboration by analyzing the actual collaborative and integrative activities of developers spanning more than one FLOSS project.

RESEARCH SAMPLE AND DESIGN

The research design and data analysis is described in the light of the research goals: sampling, initial coding and working constructs, and final coding. Table 15 summarizes the research process and an overview lists and briefly describes the sources and data that informed this study. In the first step of the analysis, the project sampling was subjected to an exploratory data collection because the organizational and technological environment for this study had to offer prospects of filling the theoretical gap (Lincoln and Guba, 1985). Thus, the coding, framing, and data analysis overlapped in time and evolved as the understanding of collaboration improved.

A group of collaborating sets of projects was sampled to enable a study of technical collaboration among developers across FLOSS projects. At the outset, three candidate environments were scanned: an integrated development environment (Eclipse), a Linux distribution (Debian GNU/Linux), and the Free Java runtime environment. All of these technical environments consisted of dozens of FLOSS projects that were collaborating at some level.

Three aspects led to the runtime environment being favored for the final coding. First, the copyright for much of the Free software written for the runtime environment projects was owned by the Free Software Foundation, which required written copyright transfer statements from the authors. In this hacking culture, people tended to use their real names when posting on mailing lists and communicating publicly. This characteristic helped to identify the authors across projects. Second, the Free Java runtime environment allowed the observation of a current, large-scale integration effort between the GNU Classpath and Kaffe projects, which proved very accessible through a friendly developer community. Third, Eclipse built on governance structures that seemed unusually complex and explicit for open source projects. The structures helped to balance the interests of the open source project and the consortium of firms that contributed to the technology.

Free Java

Java is an object-oriented programming language developed by James Gosling and others at Sun Microsystems. It was first released in 1996 and the Java trademark is owned by Sun Microsystems. Since its release, Java has gained popularity, particularly for application development and web-related programming, and many software engineers use it. Over the years, developers have acquired skills in writing Java code, and proprietary as well as open source software is being written in Java. At the time of this writing, Sourceforge.net listed more than 40,000 FLOSS projects written at least partly in Java. The official Sun Microsystems Java components impeded the basic freedoms associated with FLOSS and trapped developers who wrote FLOSS using the Java language (Stallman, 2004). The so-called Java Trap motivated developers to write a 'free stack', i.e. the sum of underlying programs needed to run a Java program. One of the efforts by Free Java developers to integrate Free Java with the GNU Linux desktop and with software from the Apache group became known as Apache Harmony (Wielaard, 2006). The integration led to a wide acceptance of Free Java but also raised new questions about the compatibility of the different Free and Open Source software licenses. In November 2006, Sun Microsystems released their Java implementation under the GNU General Public Licence (GPL), thus making it Free software.

It is important to note that, since the invention and initial development of Java by Sun Microsystems, an environment of software products, both proprietary and Free and Open Source, has developed that extends and continues to extend the Java platform in various ways. The projects in this study each provided Free components or versions of a platform or runtime environment to enable programs written in the Java language. At the same time, the Free components attempted to improve the proprietary system and expand its functionality, speed, usability, and compatibility to other Free and proprietary systems, such as Microsoft's .NET framework.

A program written in Java programming language is first compiled into Java's intermediate language, Java bytecode. The program is then executed on a Java virtual machine which interprets the intermediate language on a target system or computer. The Java platform, or runtime environment, as the system of components needed to successfully run a Java application is

known, consists of three core components: the Java virtual machine; the class libraries; and the compilers. Other components of the Java platform include tools, toolkits, and interfaces. The virtual machine is the program that executes the same bytecode program on many different platforms, devices, and hardware systems. The class libraries contain reusable code and functions that facilitate programming and enable the system independence of Java by providing basic functionality without relying on the underlying operating system. The compilers transform human readable and writable Java source code into Java bytecode and into machine code.

The example of the compiler shows how the functionality provided by Sun Microsystems is being altered and extended in important ways. One compiler turns the Java source code, as written by a programmer, into Java's intermediate language. Another compiler translates from the intermediate language into the so-called native language of the target system (machine code) usually at runtime, while the program is being used. The FLOSS compiler studied here, the GNU Compiler for Java (GCJ), can do either, plus compiling Java source code directly into machine code, thereby creating new options for users. Meanwhile, compilers have been created for other programming languages such as Python or Ruby to compile or interpret these languages into Java bytecode in order for programs written in these languages to take advantage of the Java platform. Thus the basic Java technology with its specifications and reference implementations functions as the baseline for innovations that enable an ecosystem of software projects.

The core sample studied here contains three FLOSS projects aimed at producing one of the core components of a Java platform: Kaffe is the Free virtual machine, GNU Classpath the class library, and GCJ the compiler. None of those programs stands alone, they need to be integrated to function as a runtime environment. However, for each of the programs, there were Free alternatives. GCJ implemented its own class library (libgcj) before deciding to merge it with GNU Classpath. GNU Classpath, in turn, could and had been used with a number of other virtual machines, and Kaffe could select between existing class libraries. A number of other components of the Free Java platform could be associated with the core sample because they contributed or built on resources and code from the core projects studied. About thirty projects included software for testing (the Mauve Project), other virtual machines (such as Jikes RVM or IKVM.NET), or academic research projects.

Research Process

The nature of this study is exploratory and inductive in order to construct a theory that is directly grounded in data. Many terms used in this study are defined explicitly because their meaning varied between the literature and the informants' use in the field. During work with the field data, it became clear that the explanations for observed behavior in particular emerged from a negotiation of meaning between the observer, the author, and the data (Lincoln and Guba, 1985), a difficulty that was mediated partly by the wealth of data and the multiple sources employed (van Maanen, 1979). Daily immersion in the field during the months of February to April 2005 followed the method of netnography (Kozinets, 1999; 2002). Reading all the mailing lists run by and pertaining to the three focal projects (12 lists in total), and studying weblog entries, news postings and code documentation, generated an approximate understanding not only of the technology but also of the efforts incurred by the developers during that time. Certain development issues could be tracked in real time and the field notes taken during this period informed the targets for and content of the subsequent interviews. In addition to the conversational and communicative data drawn from interviews and communication archives, the technology itself could be scrutinized by the author.

The analytical process to validate the constructs and patterns in the data followed several cycles of verification to extract meaning and condense the observed patterns, and to exclude an observer bias as thoroughly as possible (Eisenhardt, 1989). Systematic checks between interview statements and software change logs allowed for a certain level of triangulation, which was further improved by applying modest quantitative techniques of counting technology change incidents and interactions between individuals and groups of individuals (Jick, 1979). As part of a member check, all direct informants received the opportunity to comment on the full set of results from this study, and received a late working paper of this text as an email attachment.

Six primary sources of data informed this study: interviews with developers; communication archives such as mailing lists; software code; change

logs and version history archives; web-based documentation material (hacker guides, code documentation, links); and weblogs (individual and group). The interviews were semi-structured and lasted between 35 and 90 minutes, the average was around 45 minutes. Fourteen conversations were carried out by telephone (nine of which were later transcribed verbatim), two were conducted face-to-face, and seven took the form of email conversations that spanned more than one week. The language was either English (14) or German (9), and 11 respondents were located in Europe, 1 in Japan, and 11 in the US and in Canada. The interviews took place between February and June 2005. Mailing list archives were used to learn about the content matter, follow the authors involved over time and across projects, triangulate interview statements, and track technical decisions back in time. All three focal projects used mailing lists both to discuss the development as well as to communicate code patches directly when they were added to the code repository. The software source code of two projects was downloaded into a local database and coded for authorship analysis. The logged source code changes could be accessed through the research database or online, as all three projects allowed web-based access to the code repository. Anonymous viewers of the code could browse the file tree online and access or download all the files. Some of the log files were made available more conveniently through direct links on the projects' internet pages. The change logs appeared in the file structure in full length, sometimes spanning several years and listing thousands of logged changes. Additionally, the project maintainers announced an edited, shorter change log which contained excerpts of the original change log. Release announcements contained what the maintainers considered to be the more important changes. These announcements were used for the final coding of code sharing incidents.

The project web pages usually contained diverse resources for current and prospective developers, including software documentation and introduction, historical explanations and developments, links to related projects, 'hacking guides' and more. These resources helped to locate a project's role within the technical environment and the developers' goals regarding collaborations and affiliations with other projects. The hacking guides and project histories explained to new contributors the histories of collaborative efforts and the intricacies of contributing to shared code. More current issues and events could be found on weblogs[59] where developers exchanged their views on the evolution of the field and shared technological novelties that might be of interest to developers of related projects. Internet search engines helped in locating messages in mailing list archives: Gmane, Google and specialized archiving sites facilitated the recovery of list messages that were up to seven years old. All these data sources were consulted regularly and frequently in order to learn about the Free Java environment and the roles of the different technologies within the environment. The coding activities involved the above data sources in alternating intensity as the picture of the various collabora-

Steps	*Question*	*Method and Data*	*Result*
1	Is there evidence for technical collaboration across projects?	• Scanning of different technical environments • Interviews • Archival data	Sample
2	Which type of collaborative activities can be observed?	• Coding of release notes and change logs • Archival data (project communication and documentation) • Interviews	Overview of collaborative activities that span two or more projects
3	What are the patterns in the observed collaborative activities?	• Coding of authorship, contributions, and affiliations • Source code • Archival data (mailing lists and logs) • Interviews	• Long-term interaction • Frequent communication • Distributed systems integration
4	How can the observed patterns be explained?	• Coding of integration tasks (e.g. interface building) • Archival data (mailing lists and source code change logs) • Interviews	• Distribution of effort • Role of uncertainty • Communication and access to expertise

Table 14: Summary of the research process

[59] A frequently cited weblog in the Free Java world is Planet Classpath: http://planet.classpath.org

tive efforts across the projects emerged. However, only the key intermediary constructs are mentioned below and only the final coding is described in detail.

Data Analysis

Initial coding in step 2 allowed a first overview of collaborations carried out on different levels and with varying impacts. The interviews were transcribed during the first few days after the conversations and coded in the form of a simple list that sorted and counted the key themes across the interviews. This list was amended and extended by reading through the full set of interviews at the end. The focus on three central projects within the Free Java runtime environment (GNU Classpath, Kaffe, and GCJ) revealed collaborations that spanned and went beyond these projects, and differed in their technological and organizational scope.

An initial classification of activities, based on the coded interviews, led to two groups of emerging constructs: shared use, merger, reuse, fixes, and resynchronization on one hand, and fork, incubator, and platform technology on the other. Entanglement became tangible in step 2: developers identified so closely with the code they developed that they created, in words and action, organizational boundaries for their work: the social (organization) and the material (code) appeared as entangled (Orlikowski and Scott, 2008). An experimental branch organized discussions about its meaning and promise for the system and the classification had to pay careful attention to all cases in comparison. These constructs appeared as patterns in the data due to the frequency of use by the respondents. They reflected the wording of the respondents, who drew organizational boundaries when pointing to collaborative activities. The boundaries were not always obvious. An incubator project, for example, referred to a technological experiment with speculative outcome. Organizationally, it could be either a fork or an experimental branch within the same project. The terms and constructs are explained in detail in Section 4.

The third step in data analysis focused on the organizational consequences of collaboration. The collaborative activities broadly varied on two dimensions, their organizational and their technological impacts. These dimensions were used to structure the final coding. First, the organizational dimension distinguished collaborative activities that formed new organizations from those that connected two or more existing organizations. It became clear that forking and experimental branching were an integral part of technology development in the Free Java runtime environment. Hence, research on collaboration across FLOSS projects would have to take into account the fact that the foundation of new organizations could establish a collaborative activity. Additionally, forming new organizations was clearly seen by the informants as playing an important role in the development of the technology.

The second dimension discriminated according to technological impact. The subset of collaborative activities resulting in considerable technological impact was followed in time and coded with regard to authorship and cross-affiliation of the authors. The project maintainers announced new releases to the project. They communicated the major changes to the technology featured in the current release. Their choice of changes worth communicating explicitly skimmed for the more important changes among the hundreds of changes made to the technology during the time that had elapsed since the last release. The analysis considered only the collaborative activities among the code changes communicated in the releases. The maintainers usually credited the contributors individually and thereby carried the responsibility for not forgetting contributions that individual developers may consider important. It could not be a spontaneous, arbitrary choice to leave out important contributions, except maybe the maintainer's own. The complete change logs and the interviews suggested that the changes not reported in the announcements included minor improvements to the performance of the software or bug fixes. Hence, they were not analyzed in detail.

The set of activities that underwent detailed coding was split in two according to the formation of new organizations: the complete, recent release announcements of all of the three focal projects were coded separately from the forks spinning off the three projects. The announcements over a period of six (GNU Classpath) and 18 months (Kaffe and GCJ) amounted to 158 technology changes in total. The three projects followed different release schedules, which meant varying the observation periods in order to have a balanced sample of recent changes. GNU Classpath released about every two months, whereas Kaffe only announced one development release in

2004. GCJ was part of the larger GNU Compiler Collection (GCC) project and aggregated its relatively rare GCJ-specific release announcements over the branches depending on their progress and the need to communicate. Coded together, the observation period of GNU Classpath was cut to six months to contain a number of changes comparable to the Kaffe project (72 for GNU Classpath and 76 for Kaffe). GCJ only announced 10 changes during the 18-month period.

Technical changes stemming from collaborative activities were separated from technical changes stemming from contributions. Fifty of the reported 158 changes were directly related to a collaboration with another project. In these cases, code was imported from another project or jointly used with another project. Through mailing list searches, the technical changes could be attributed to the authors or the group of authors that implemented the change. The 50 collaboration incidents marked a lower bound since it was possible that more changes could be used jointly by other projects but were not reported as such. The analysis followed the collaboration incidents more closely and tracked the author's affiliation with the project back in time. Additionally, the references to mailing list discussions were recorded and unclear issues marked and later discussed in interviews with the authors or core developers of the corresponding, importing project.

The new organizations (23 forks) that were directly linked to by the three focal projects were coded according to founding date, their purpose with regard to eventual research output (publications), the author's overlap and affiliations with the old organization, and possible back-merging activities. Again, the number of these organizations marked a lower bound as possibly more forks existed but did not report their use of the original project's code.

The analysis then sought to identify explanations for the observed behavior. The interviews were coded in terms of the rationale given for collaborative activities. The interview guidelines included questions about the nature of cross-project collaborations, examples of specific activities that the informant was personally involved in, as well as why-questions that did not aim at personal motivation but at the specific reasons for why an activity was called for in technical terms. As mentioned above, the interviews were used to discuss changes observed in the code and also to follow the histories of specific implementations, particularly of interfaces.

RESULTS

Early on during the interviews and the analysis of the code changes, it became clear that both the organization of the projects and the technology were open to new participants and receptive to changes originating from other projects. FLOSS projects formed around a specific agenda to build particular software, and a set of goal and mission statements about what that software should do, to whom it should be useful, how it may be used, and so on. Since the software is Free or Open Source, it may be viewed, downloaded, and changed by anyone. Developers and users sent back bug reports and suggested fixes, contributed to the discussions on the mailing lists, or altered the technology in significant ways. In the projects analyzed here, the users' comments and contributions were usually solicited and welcomed explicitly.

At every moment in time, a project could be defined as a group of individuals working on an emerging code base that makes up a software product or designs a software product in the making. Working meant developing and discussing development. Respondents understood project boundaries as technical rather than social although they repeatedly voiced concerns about this separation as being artificial or historically determined. If a user and developer of a virtual machine contributed a security algorithm to, say, GNU Classpath, the contribution was credited to the user and labeled as shared code or as a merger because the algorithm was not exclusively developed for GNU Classpath, but reused from the virtual machine. If the user contributed regularly, he could receive write access to the source code. Even then, with the committer and the author being the same person, a contribution could be considered shared code simply because it had been developed mostly outside of the project.

Figure 9 displays a few sample transactions involving five projects, two of which belong to the core sample (Kaffe and GNU Classpath). In the environment studied, developers engaged in various collaborative activities. The Jessie project reused security software developed in GNU Classpath and the code was later merged back and then reused in its updated form by GNU Classpath developers. Kaffe merged its networking classes with those developed for GNU Classpath. This was a long-term process that involved many individuals and represented one of the core integration efforts between the two projects. Janos VM and JC spun off as friendly forks from Kaffe and GNU Classpath, respectively, Janos in 1999, JC in early 2004. Each built on the code base of the original project but maintained collaborations with developers from the other projects. In 2005, Kaffe merged back JIT3 code from the Janos VM fork.

Figure 1 cannot show the collaborative groundwork within a technical system. The focus on important technical changes tends to forget that users and developers regularly and frequently discussed and fixed minor problems across the projects that made up the Free Java runtime environment being studied. Informants for this study reported applying fixes and improvements to other projects, that is to software of other development organizations, as part of their regular development work (developers P, K, A, and M).

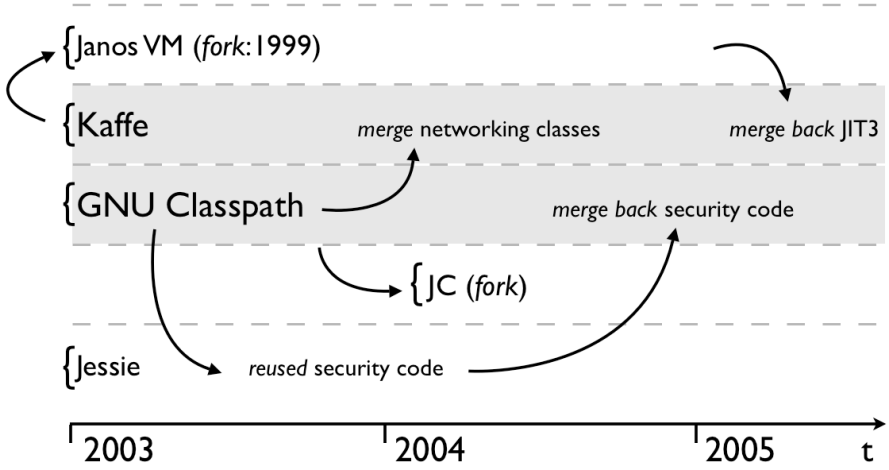


Figure 9: Examples of integration activities across organization boundaries

Integration through shared code: transaction type one

The shared use of software code in the sample relates to long-term collaboration of developers across projects. While the data did not reveal integration performance, the software products studied, and the products of the core projects in particular, formed a functioning runtime envi-

ronment which required a certain level of integration. Each release communication coded for the analysis tracked integration performance by referring to successful tests and merged code. Collaboration activities were thus aimed at the long-term goal of successful technical integration and occurred in parallel to a social integration that started to unite members of the core projects in the sample.

Analyzing the technical change announcements of the three focal projects revealed that 50 out of 158 announced changes originated from outside the respective projects. Hence, almost one third of announced technical changes were related to incidents of code sharing. Table 15 summarizes these findings. According to the announcements, 46 technical changes were fixes or improvements to existing code. If these important but incremental changes are subtracted, 45% of the announced changes originated from outside the project. This classification of technical changes was informed primarily by the announcements, and further by authorship analysis and the discussions and documentation that historically surrounded the technical changes. Table 16 lists the 50 changes originating from outside the respective projects with their source and the original classification.

Project	*Announced changes*	*Original contributions*	*Code improvements*	*Shared use of code*
GCJ	10	6	0	4
GNU Classpath	72	38	22	12
Kaffe	76	18	24	34
Total changes	*158*	*62*	*46*	*50*
Percentage		**39%**	**29%**	**32%**

Table 15: Origin of technical (source code) changes

Where applicable, the committer and the author of the change were recorded and the corresponding email conversation logged. Of the 50 changes originating from outside the project, 42 were authored by a developer who was active on the importing project for at least one year or, in cases of co-authorship, came from a project in which at least three developers were active on the importing project for at least one year. The time period of the collaboration relative to the implementation and relative to the announcement of the changes varied broadly across the sample. In the eight remaining cases, the overlap could not be identified or was negligible, for instance amounting to a handful of posts by the original author on the importing project's mailing lists. Cases where a single author could not be identified concerned the sharing of classes from GNU Classpath (into Kaffe or GCJ). Twenty-six out of the 50 imports originated from one of the two other projects in the sample. Despite certain overlaps, it seemed reasonable to keep all the changes in the analysis. What was labeled as an important change in one project was not necessarily announced in the other project. Because the attribution of relevance was determined by the importing project (see section 3.3), the analysis had to include each change instance according to this design. Canceling out the change instances would assume that the projects in the sample were somehow related ex ante.

The explanation for sharing code with another project was to prevent duplication, or 'reinventing the wheel' (Developers K, A, D). The origin of the decision to share a component was difficult to track because, just as in other FLOSS projects, the implementation counted more than the intention (Mulgan et al., 2005; Raymond, 1998), and developers tended to act rather than discuss. Unilateral sharing activities were rare: for six out of the eight changes without long-term collaboration, no collaboration could be found at all.

Shared code is usually clearly marked as such, either in documentation or in mailing lists. For the shared code between GCJ and GNU Classpath, developer P explains the coordination process:

> *The two projects are overlapping, GCJ builds on Classpath at this point. The library part of GCJ is Classpath. Whether someone works on the library part of GCJ or Classpath is kind of immaterial, the code goes both places to the extent applicable. The current GUI [graphical user interface] work, SWING, and AWT goes into both Classpath and libgcj and it's the same source code. And there's policies, if you make a change you're supposed to check it in both places at the same time or at least into Classpath first. so it'll eventually get merged into GCJ. (Developer P)*

There were shared task lists and agendas, but only a few developers regularly used them to communicate what tasks they were working on. Others mentioned their current activities on Internet Relay Chat (IRC) or in the mailing lists. The announcements then triggered discussions among users about technical consequences for other projects and users. As IRC was not archived by the projects studied, the developers felt the need to document certain decisions for new participants. A wiki in GNU Classpath listed a series of recent coding decisions relating to collaborative activities.

Two core developers in the sample seemed to assume a particular responsibility for bringing the technologies closer together: developer K between GNU Classpath and GCJ, and developer D between Kaffe and GNU Classpath[60]. Their efforts involved careful monitoring of changes to the respective code bases (source trees) and communication about the intended and ongoing sharing activities. This observation corresponded to the finding of specialization in other open source projects (von Krogh et al., 2003).

Project (Target)	*Release*	*Patch / Feature*	*Source*	*Classification*
GNU Classpath	0.14	javax.awt.imageio support	gdkpixbuf	reuse
GNU Classpath	0.13	The http url protocol handler has been replaced with a full HTTP/1.1 version	GNU inetlib	incubator merge
GNU Classpath	0.13	A new ftp url protocol handler	GNU inetlib	incubator merge
GNU Classpath	0.13	The java.util.Locale support is now based on the Common Locale Data Repository (CLDR) Project (see http://www.unicode.org/cldr/).	CLDR Project	reuse
GNU Classpath	0.13	Added implementations of javax.xml (JAXP 1.3),	JAXP 1.3	reuse
GNU Classpath	0.13	org.xml.sax (SAX2)	CPX	indirect reuse
GNU Classpath	0.13	java.awt.Robot support with GdkRobot in the gtk+ awt peers.	gdkRobotPeer	reuse
GNU Classpath	0.12	A much improved version of X.509 certificates has been added, including a robust certificate path checking algorithm. Also included is an implementation of the RSA signature scheme.	Jessie	back merge
GNU Classpath	0.11	New javax.crypto, javax.crypto.interfaces, javax.crypto.spec, javax.net,	GNUcrypto and Jessie	merge
GNU Classpath	0.11	AWT GDKGraphics scaling.	libgcj	merge
GNU Classpath	0.11	New configure flag --enable-gtk-cairo to build Graphics2D implementation	gcj	merge
GNU Classpath	0.11	GNU JAXP	JAXP	reuse
Kaffe	1.1.5	* Resynced with latest GNU Classpath.	GNU Classpath	merge
Kaffe	1.1.5	Added GNU EmbbededWindow extension from GNU Classpath. gcjwebplugin is known to work with the extension	GNU Classpath	merge
Kaffe	1.1.5	GNU Classpath's implementation of AWT/Swing has been merged.	GNU Classpath	merge
Kaffe	1.1.5	New Nano-X AWT backend.	Nano-X	reuse
Kaffe	1.1.5	Merged in JIT3 for powerpc from JanosVM.	JanosVM	merge
Kaffe	1.1.5	KJC temporarily replaced by jikes.	Jikes RVM	reuse
Kaffe	1.1.5	DNSJava merged	dnsjava	reuse
Kaffe	1.1.5	Jessie merged	Jessie	merge
Kaffe	1.1.5	JZLib merged	JZLib	reuse
Kaffe	1.1.5	gjdoc merged	Cp-tools	incubator merge
Kaffe	1.1.4	GNU Classpath merges	GNU Classpath	merge
Kaffe	1.1.4	Object serialization	GNU Classpath	merge
Kaffe	1.1.4	almost all of java.io from Classpath	GNU Classpath	merge
Kaffe	1.1.4	java.nio, java.net	GNU Classpath	merge
Kaffe	1.1.4	java.util, java.util.regex	GNU Classpath	merge
Kaffe	1.1.4	java.math, javax.naming	GNU Classpath	merge
Kaffe	1.1.4	java.text, java.beans	GNU Classpath	merge
Kaffe	1.1.4	java.net.protocol.file.Handler	GNU Classpath	merge
Kaffe	1.1.4	java.util.Random	GNU Classpath	merge
Kaffe	1.1.4	java.awt.GridBadLayout	GNU Classpath	merge
Kaffe	1.1.4	java.awt.geom	GNU Classpath	merge
Kaffe	1.1.4	javax.swing.event.EventListenerList	GNU Classpath	merge
Kaffe	1.1.4	javax.swing.text.AttributeSet	GNU Classpath	merge
Kaffe	1.1.4	Updated sound code from Tritonus.org	Tritonus.org	merge
Kaffe	1.1.4	Updated javax.net.ssl/javax.security code from Jessie	Jessie	merge
Kaffe	1.1.4	moved automake scripts and m4 files out of top directory	ALSA.CVS	reuse
Kaffe	1.1.3	java.beans	GNU Classpath	merge
Kaffe	1.1.3	java.util.Date	GNU Classpath	merge
Kaffe	1.1.3	java.util.jar	GNU Classpath	merge
Kaffe	1.1.3	java.net	GNU Classpath	merge
Kaffe	1.1.3	java.io	GNU Classpath	merge
Kaffe	1.1.3	javax.swing.EventListenerList	GNU Classpath	merge
Kaffe	1.1.3	Resynced with existing Classpath, GNU JAXP, Jessie.	Jessie	merge
Kaffe	1.1.3	Further merge with GNU Classpath: Collections, many networking, IO and zip classes.	GNU Classpath	merge
GCJ	GCC 4.0	classes in javax.xml, plus updated versions of org.xml.sax and org.w3c.dom.	JAXP	merge and reuse
GCJ	GCC 4.0	new javax.crypto, javax.crypto.interfaces, javax.crypto.spec, javax.net, javax.net.ssl, javax.security.auth, javax.security.auth.callback, javax.security.auth.login, javax.security.auth.x500, javax.security.sasl and org.ietf.jgss packages from the latest GNU Classpath 0.11 developer snapshot release	GNU Classpath	merge
GCJ	GCC 4.0	support for java.util.regex	GNU regexp	merge
GCJ	GCC 3.4	call graph optimization code improves gcj's method inlining abilities with support for out-of-order methods and inter-class method calls, along with improved code size heuristics. It is enabled by default when compiling at -O3.	Hubicka	reuse

Table 16: 50 changes originating from outside the target project with reuse classification

Monitoring merging activities consisted of comparing the shared software components, and working with others to eliminate any differences between the original and reintroduced implementations, since changes made in one project could make the code incompatible with the other pro-

[60] Upon later request, developer D indicated that he did not get paid for programming in Java nor for the integration work at the time of this study (2004 and early 2005). Developer K did not provide information.

ject. Additionally, a few important pieces of the shared code (in the example of GCJ and GNU Classpath the java.lang.string class) needed to remain specific to each project for historical reasons or due to optimization and performance issues. Developers K and D devoted considerable resources, possibly the majority of their contributions, to integration activities. They worked without formal authority nor could they necessarily be considered central figures in their respective communities.

For important technical changes, developers shared components with other projects and continued to contribute to both projects. The observed practice could have been transient and technology-based only. The ability to cheaply identify and access technology might be independent of the organization of technology development. However, the data from this sample revealed that not only did developers work with the software of other projects over long periods of time, they also interacted with the developers from other projects and built social ties that linked the projects. The Apache Harmony effort published in May 2005 represented an institutional move that tied the projects closer together after years of social interactions on the level of code (Wielaard, 2006).

Transparency, the frequency of communication, and long-term, multiple project affiliations by developers seemed to enable distributed integration through shared code. First, populated mailing lists provided a resource for new joiners and users to receive technical help and to learn about the technology and its development (von Krogh et al., 2003; Lakhani and von Hippel, 2003). The continuous work of maintaining and adding compatibility and developing standards needed to involve users from various technical backgrounds. Developer M specifies the forms of interaction on the GNU Classpath mailing list as follows:

> *There are two kinds of interactions, one is [the users] explicitly ask for new interfaces and features: "We want to use it in this way. is this possible?" And most of the time it's just splitting what you have now in an extra direction. That's what you do with software. You make certain functionality depend on input from the user. The other thing is, they say, "we have to do it this way because you defined these interfaces and we don't actually get them". Mostly, it turns out that you made some design mistakes or, at least, you didn't document them very well. [...] Users on our mailing list mostly define our interfaces. And we're there, of course, and act like we know a lot. And we do because we saw a lot of users. The good interfaces are when two of your users duel out what they need with you present.*

As this quote illustrates, the joint use of mailing lists served a number of purposes. The evolution of interfaces was a continuous, social process. Notably, the details of how the system components actually interoperated were subject to constant improvements and negotiation. These negotiations focused on new and emerging features but extended to the improvement of existing functionality, such as new graphic support tools and components that allowed compatibility with other platforms or distributions. Note that the interfaces were built jointly by developers of several projects and not only by one party that sought compatibility with another technology.

Second, a constant flow of communication, both via mailing lists and IRC, allowed the developers in the sample to discuss what they were working on and sort out technical problems. A few developers reported being present on several IRC channels (of two or more of the projects in this sample) more or less all day long (K, G, D, M, A, F). These channels provided an important social meeting opportunity along with the privacy to voice opinions which were not appropriate for the public domain (developer K). They were also testimony to the social ties that developed between many of the project developers during the two years that preceded this study. Developers D (Kaffe) and M (GNU Classpath) reported having initiated regular face-to-face meetings in 2003 for developers of both projects. Also, the Planet Classpath weblog attracted contributions and comments about new technical breakthroughs as well as personal opinions and confessions. Ideas and achievements were part of this blog as well as accounts of personal events, such as the life-threatening illness of one developer, which triggered consolations and personal contributions by members of the wider community.

Third, as observed in other FLOSS projects, most tasks were self-assigned by developers (Yamauchi et al., 2000; Haefliger et al., 2008). One consequence of self-assigned tasks in convergence is a time lag between a change being introduced from one project and developers in the other

project intervening for compatibility. This effect applied to corrections as well. With code that served as a core system for other software solutions and ran on dozens of platforms, such as both Kaffe and GNU Classpath, sorting out the side effects slowed development and created time lags:

> *Often you have to adapt the patches [when sharing code], which takes time. You have to count on lags of up to several months because nobody dares to rewrite the patch. (Developer K)*

In order to accommodate time lags, developers tended to stay active and oversee future developments of their code after it became shared code among the projects. The shared expertise led to mutual help and learning.

Back-merging, the reuse of shared components that had been improved outside the original project, was given great importance in the developers' evaluation of collaborations. Back-mergers could be observed in at least three cases and were included as instances of code sharing. A number of specific candidate technologies for back-merging, both back into Kaffe and back into GNU Classpath, could be tracked in mailing list discussions and task lists. The perceived importance of back-merging clearly motivated long-term collaboration and created incentives for a general convergence between the projects. If code was being shared among two or more projects, improvements to the code were shared as well. Hence, whoever contributed to the code, both projects would benefit from the improvement. If this process worked properly, back-merging should remain incremental and few important technical changes should stem from back-merging. This corresponded to the actual observations.

In summary, integration through shared code was frequent, without intervention of a central system integrator, and enabled by transparency and frequency in communication and long-term affiliations with multiple projects. Collaboration exposed developers to new technology, which in turn impacted on their previous work. Collaboration was often motivated by gaining access to a solution that would have had to be written from scratch otherwise. By collaborating with another project, developers (A, G, K, D) sought to avoid duplication. However, the shared code had to be adapted to new users and new environments over time, and it had to be maintained. A shared responsibility and common knowledge about the history of both the code and the collaboration created identification across the projects. This identification was explicitly promoted by at least two developers of Kaffe and GNU Classpath, who organized regular face-to-face meetings. Several developers referred to the group of projects, spanning beyond but including the sample studied here, as one family (developers P, K, M, D).

Integration through branches and friendly forks: transaction type two

An evolving project needs mechanisms for exploration of new ideas and functions as well as exploitation in the sense of serving current user needs. Branching and forking allow such exploration to take place. Thus, the branches and forks can be seen as a part of an extended family of integrated software products. Branching of the code base builds on the infrastructure of an existing project, forking establishes a new project based on code taken from an existing project. A branch is necessarily collaborative as the author of a branch makes use of the given project infrastructure, while a fork does not have to be collaborative, but was observed in this study as only taking place under friendly auspices. The following quote by a core developer of Kaffe introduces the issues:

> *Usually, we think that forks are something bad. But what people tend to overlook is that [forks] have great advantages, because projects may develop in multiple directions. Some can or want to do only their thing without paying attention to others. And most of Kaffe's forks are in research: Janos, Jessica, Hydra, Latte. They prove, invent, and implement new and exciting things without arguing: „Why should exactly this patch...?", and so on. A project with the ambition to be portable [such as Kaffe] spends a lot of time identifying and fixing small problems. Developing for radical problems becomes difficult in the main branch. [...] And research needs a solid basis, which we're offering. That's why I think forks make sense. They avoid social tension from the outset. (Developer D)*

The Kaffe and GNU Classpath projects linked to a total of 23 forks on their websites between 1998 and 2004. These projects were based on, or relied on, a derivative version, or parts of the original code at the time of the fork, and formed their own development projects. The term fork often carries a negative connotation (Raymond, 1998) as a schism that tears a project apart and creates social distress. In this environment, however, no hostile project separations were observed. However, this is only a casual observation and could not be tested, as hostile forks might not have been reported or were intentionally not mentioned.

Uncertainty and lower development costs related to negotiation and communication appeared as reasons to branch or fork code. Negotiation and communication costs occurred whenever a large group of users behaved as a constituency to the software. The costs accrued independently of the actual problem solving, fixing, and development work before and after intervention by a user. Avoiding interaction with the large group of users in favor of an emerging, smaller group of users created incentives to fork. Friendly forks cater to the heterogeneous needs of users (Franke and von Hippel, 2003) while the forked project, the solid basis, benefits from network effects (Economides, 1996) due to increased portability and the larger user base of its code. Thanks to the friendly fork more users and developers will be acquainted with the technology and refer back to the forked project. The work in the project was viewed as costly because it triggered tensions and led to negotiation costs about technical decisions. Branches and forks offered an efficient way for mediating these costs while working with the same technology.

Both GCJ and GNU Classpath entertained development branches.

> *GCC has two types of branches, release branches, and development branches, where you make work that's larger or more speculative. It's just a way to let you do major, possibly disruptive changes without disturbing anybody who's not interested in them. So GCJX falls into that category. It's not complete, doesn't fully work yet and because it's a replacement for GCJ, making it all work entailed on the branch deleting the old GCJ. I couldn't do that in an environment where other people are using it. I couldn't do it until it's done, and that's still months away.(Developer T)*

Developer T, who was the principal author of the GCJX branch, used similar arguments as did authors of forks. Branches offered the same advantages as forks, but inside existing projects, which meant that the communication and source code management infrastructure did not need to be replicated. Additionally, a branch was regarded as the seed of future major changes to the existing project, even a replacement. In the case of GCJX, the replacement was deemed necessary due to significant changes in the specifications of version 5.0 of the Java language. A development branch of GNU Classpath followed the same objective of adapting the project to implement Java 5.0 specifications in parallel.

Similar to the integration through shared code, the enablers of integration through branches and friendly forks included transparency (openness), frequent communication, and long-term, multiple project affiliations by developers.

First, research projects played an important role as authors of forks because researchers appreciated the openness provided by the FLOSS licenses. Publications documented 16 out of the 23 forks. The relevant working papers, theses, or conference proceedings were available on the projects' websites or on a university site that was linked to from the projects' sites. Some of the forks were initiated by research teams, which published several works over the years. The 16 publicized forks did not correspond to the 16 cases of long-term collaboration (see below). Only in one instance could neither a publication nor long-term collaboration be logged. The importance of researchers as authors of FLOSS forks did not seem to be an exception in this sample. The Jikes RVM, an open source virtual machine that was initially sponsored by IBM, documents over 100 publications related to the project (see Alpern et al., 2005)[61]. In the case of Kaffe, developer C and a team at university built their project on the basis of Kaffe, which included changing some of the basic internals of Kaffe in order to demonstrate a new approach to object-oriented programming.

> *I'm using Kaffe because it's concise, because it's a simple implementation with which we could quickly demonstrate an idea. For research pur-*

[61] For more information on the Jikes RVM turn to: http://jikesrvm.sourceforge.net/

> *poses, the most important thing is that we can build a prototype that proves that the idea works. We don't need commercial quality that works under all circumstances but it has to function. And at that time, I found the option very attractive that someone may later reuse our work. (Developer C)*

The licenses provided by Sun Microsystems for Java products tended to offer source code access to researchers more easily than to other industry partners. But since Java products were not open source at the time of this study, the researchers could not distribute their adapted versions publicly.

Second, on the mailing lists and on weblogs, developers within the Free Java environment discussed not only current features and project details but also the larger picture of the technological environment. Discussions conducted with particular verve involved technological visions and breakthroughs to other language environments, and the general evolution of the Java language. Excellent examples were the enthusiasm for the IKVM.NET project[62], or the broad interest in the Apache Harmony[63] effort to unify or integrate the various efforts to create a compatible, open source implementation of the Java 2 Standard Edition (J2SE) runtime environment. The awareness of the wider evolution of technology implied an appreciation of technical changes in the environment. This appreciation included not only an interest in new features of closed projects (closed in the technical sense), but also a sense of the uncertainty involved in new ventures. Developers who had been implementing Free Java components for up to ten years shared a close understanding of technical boundaries to other languages and environments. Uncertainty was mentioned repeatedly (Developers T, P, K, D) as the reason why a project forked the original project's code. To the developers, it often seemed obviously impossible to incorporate the vision and goals of the forking project into the original software, thus founding a new project was the logical consequence. In a separated environment, where radical changes to the code did not affect the original code and its users, the developers could follow more experimental approaches and speculative ideas.

Third, developers from 19 out of the 23 forks maintained a long-term collaboration with the originating project. Developers of 16 forks frequently contributed for at least one year. For two additional forks, the analysis revealed a close involvement with the originating project for at least nine months, and the author of one fork continued to contribute infrequently to GNU Classpath for more than 18 months. The four remaining forks did not generate long-term collaborations. For two of them, contributions by their developers amounted to a mere handful of posts on the originating projects' mailing lists, and for the other two, no traces of interaction could be identified. Similar to integration through shared code, no central authority could be found to be involved in decisions about branches or forks.

Changes in interfaces were usually up for negotiation between the external project involved and existing users (Developers M and T), and a large volume of communication in the mailing lists related to relative details compared to the important changes as highlighted by the release announcements (see also the quote by Developer D at the beginning of this sub-section). The developers discussed interdependencies between components and the consequences for their use of the software. If a change to the code affected someone's ability to use the software, the consequences were mentioned via the mailing list. The developer complained about any misfits between his or her use of the code and other software, and the problem got the attention of the project. This normal process of advancing the software created development costs, which included reading high volumes of communication traffic inside a single project. The average monthly load of messages on the main mailing list of the GNU Classpath project in the first 6 months of 2005 was 141. Considering that many developers read the lists of several projects, the reading alone consumed considerable amounts of time. Negotiations with the heterogeneous demands of users added to the costs of developing a solution. Thus, long-term, multiple project affiliations by developers enabled integration through branching and forking be-

[62] The project's site can be found at: http://www.ikvm.net/

[63] The announcement on May 6, 2005 triggered wide discussions across the larger project of projects who implement runtime components such as the projects in this sample. The announcement, and the involvement of a number of developers of the sample projects in the original announcement, reverberated on the mailing lists and on the Planet Classpath weblog (referenced above). The announcement may be found here: http://mail-archives.apache.org/mod_mbox/incubator-general/200505.mbox/%3CCA4BEB82-3D84-457D-9531-1477DD749919@apache.org%3E
See also: Wielaard, 2006

cause developers could explore new ideas and features for the entire system at lower costs while staying in touch with the original code base.

Enablers of distributed systems integration

As observed here, project boundaries allowed for partially integrated organizations that saved development costs through reuse and facilitated convergence among, both, the groups that developed the projects and the code. Roughly a third of important technical changes entered the code base through collaborations with other projects via code sharing (see Table 2), and forks appeared as an accepted and useful way to create new module boundaries and enable one module to work without the forked other but with the rest of the runtime environment (the system). Three enablers of distributed systems integration stand out: transparency (openness), frequent communication, and long-term, multiple project affiliations by developers.

First, the source code was downloaded directly from the project that developed and maintained the software, and, in addition, the developers could be reached through public mailing lists. The openness of project boundaries with respect to newcomers and their knowledge shaped the collective of projects. The projects kept their technical strongholds and responsibilities and at the same time converged towards integrated, technical systems by exchanging code and development resources in the form of individuals who contributed to more than one project. Hence, innovative activities that are usually considered to take place inside firms (Dosi, 1988) and even require the existence of firms (Ghoshal and Moran, 1996: 33) could be observed as taking place between and across FLOSS projects.

Second, the context observed here is akin to work in development alliances such as the 'stealth' bomber example described by Argyres (1999), where continued interaction and shared information systems mitigated coordination costs across organizational boundaries. Developers in the sample referred to the complex tasks of developing shared code. The complexity, in their perception, was rooted in the multiple stakes of users depending on their platform and system environments. Multiple reviews in both projects that shared code enabled a careful improvement that cycled through long discussions and technical references in order to reach a consensus among the users. Thus, the frequency of communication enabled the distributed integration of complex technology. The establishment of trust was mentioned in the same vein. Trust could ease the flow of communication as it enabled the participants to criticize more easily and express opinions that might have remained concealed otherwise. The use of IRC channels for private communication that should not be logged and published pointed to the need for "unstructured technical dialog" (Monteverde, 1995): the exchange of experience and tacit knowledge.

Third, with certain important exceptions (Dahlander, 2007; Henkel, 2006; Harhoff et al., 2003, von Hippel, 1987), firms tend to keep knowledge protected and proprietary, which leaves binary choices for collaborating partners with respect to coordination costs, and consequently access to the knowledge resources entails contracting. Since FLOSS project boundaries are less rigid, the costs of negotiating and communicating can be split among the members of various projects without formal alliances. Long-term collaboration among projects enabled the exchange of knowledge despite a separate technical domain (and organization). Again, complexity and trust were used to justify long-term interactions with users of other projects.

SOCIOMATERIAL TRANSACTIONS: SYSTEMS INTEGRATION ACROSS FLOSS PROJECTS

Developers in this sample were able to collaborate across more than one project to integrate complex software components into a larger system. The developers' behavior and rationale can be summarized in terms of two types of organizational practices and three enablers for distributed systems integration: integration through shared code and integration through branches and forks enabled by transparency, frequent communication, and long-term, multiple project affiliations.

With sociomaterial transaction I refer to the two types of organizational development practices that forge a technical system by socially and technically linking the work of many individuals. The term sociomateriality points to the inseparability of individuals and their working relationships from the artifacts, which lie at the core of the working relationships. More generally, "humans/organizations and technology are assumed to exist only through their temporally emergent constitutive entanglement (Orlikowski and Scott, 2008: 457)." The observed transactions involve developers reusing code and engaging in continuously demonstrating why it makes sense to do so, by explanation, by assuming authorship, and by extending development. In other words, the transactions describe activities that technically integrate the work of two or more groups of individuals by making their work interdependent and, thus, one group's work immediately relevant to the other. A sociomaterial transaction manipulates the social fabric of technology development by developing the technology so as to make new connections between individuals necessary and intelligible.

The analytical separation of the two activities of technology development for integration and social interaction is an artificial separation by the observer because the code that was to integrate components into systems only came to perform what it was meant to through interpretations, negotiations, and mutual agreement. Nevertheless, the separation makes analytical sense as it allows for a comparison against the baseline model informed by theory. As the analysis showed, the developers were fully aware of the economic relevance of their activities and engaged in the transactions described for economic reasons. Comparing the implications for systems integration in loosely coupled organizations with the findings of this study reveals parallels and differences as follows. The literature on organizational coupling and systems integration led to three predictions for this exploratory study of FLOSS development which can now be appraised in light of the findings.

First, and unsurprisingly, FLOSS projects rely on collaboration and knowledge overlaps in order to integrate a large technical system. Previously unconnected programmers interacted over long periods of time, as predicted by the baseline model (Brusoni and Prencipe, 2006). In addition to the prediction from the literature, the role of the users appeared as critical in defining and evolving the software interfaces within the system.

Second, and rather surprisingly, the system studied did not feature a central project dedicated to systems integration. This finding deviates from the literature of systems integration (Dosi et al., 2003), which predicts a central, indispensable role for systems integration in such an environment. Rather, a number of developers and users from different projects spent time on integrative tasks and setting up minimal coordination routines to be followed by the developers. However, the users and developers of system components seemed responsible for setting up interfaces to other components in order for 'their' component to work with the system.

Third, the baseline model led us to expect a range of organizational mechanisms to deal with uncertain developments in the components and the dependencies among the components (Staudenmayer et al., 2005). Users and developers of the Free Java runtime environment reverted to a host of activities aimed at raising or lowering boundaries between projects and between modules. All forms of code sharing observed aimed at facilitating collective use and development of code and, thus, lowered boundaries between projects. Branches and forks raised boundaries within projects or across projects where there had been no boundaries previously. The accompanying communication and collaboration demonstrated that integration continued on the system level and the boundaries set up new transaction-free zones for developers to experiment with new technologies (Baldwin, 2008).

The sociomaterial transactions describe how distributed developers (and users) define and set up component interfaces as part of a long-term

process of interaction with other developers. The visibility and accessibility of the technology affords a range of design options for sharing resources and creating interfaces between technical components. Developers engage with these options as they see fit and they carry the costs associated with the integration tasks, relying on support from and co-engagement with experienced developers. The performance of the sociomaterial transactions is most visible when users "duel out what they need" (Developer M) in front of other, more experienced, developers who help and assist if necessary. This finding resonates and adds precision to the concept of bazaar governance by Demil and Lecocq (2006) who found that coordination depends on the affordances of code released under a FLOSS license.

The question whether the sociomaterial transactions described can also be found in contexts beyond FLOSS development cannot be settled here, yet the results contain pointers. All three enabling factors seemed indispensible for distributed systems integration and withdrawing one may either bring development to a halt or jeopardize the maintenance of the software on the long run. Hence, the sociomaterial transactions of sharing and forking may be bound to environments that feature transparency of knowledge assets and allow for frequent communication between developers and their ability to stay active on multiple projects. The combination of factors might imply a process of social integration that supports and accompanies technology development. Ultimately, distributed systems integration as observed here may prove to be more efficient than central systems integration due to the distributed, self-selected task allocation. Developers work on features they perceive as useful, direct their integration efforts to where they think it is needed most, and attend to the improvements and maintenance of integrated modules out of self-interest.

Free revealing solicits development support (Henkel, 2006) and may, crucially, trigger the building of interfaces to other software. In collaboration with users, firms could accomplish the work of technical alliances between firms but at a lower cost since, under distributed systems integration, all parties contribute to building the interfaces (for a discussion of open standards, see Hyvättinen, 2006). Testing the impact of the two sociotechnical transactions on the costs of innovation would mean comparing the results of technical alliances between firms with and without user involvement at the technical level.

Ghosh and colleagues (2002) found that almost 30% of FLOSS developers in their survey contributed to more than five projects, which could indicate that distributed systems integration permits synergies between individuals and projects. Future research on user projects should investigate the role of individual mobility between projects. Contractual exclusivity or loyalty tends to tie a worker to one firm and mobility becomes a critical issue for knowledge management. Allowing developers to enact the two sociomaterial transactions, however, the threat of mobility may become a promise: if developers work for several organizations at a time, reusing knowledge across the organizations, the total amount of knowledge available to any one organization can increase. The surrounding legal policies and revealing strategies (Henkel, 2006), the accessibility and integration of the mobile individuals (Dahlander and Wallin, 2006), and the mobile workers' actual behavior, all deserve closer scrutiny.

IMPLICATIONS FOR THEORY AND METHOD

Four implications for theory can be derived. First, system integration tasks in complex, technical systems can be handled in a distributed manner, at least in FLOSS development. This finding represents a challenge to the literature on systems integration (Dosi et al., 2003; Prencipe, 2003). However, the efficiency and effectiveness of the systems integration observed was untested and should be subject to future research.

Second, if developers from several projects contribute to the building of component interfaces, the costs are effectively shared across organization boundaries. This finding adds a subtle twist to the innovation literature by demonstrating that free revealing (a condition for distributed systems integration) may not only solicit product development support (Henkel, 2006) but also systems integration support. While users are known to develop and share features and fixes to a given product (von Krogh et al., 2003; Franke and Shah, 2003), they are less known to build compatibility, that is interfaces that allow a new product to work within given technical systems or vice versa.

Third, this research contributes to a more detailed understanding of the economics of user innovation where users organize in projects to develop technologies that rely on interoperability with other technologies. Firms can learn from lead users who adapt a product according to their needs and thereby inform the manufacturing of new versions of the product (Morrison et al., 2000). In addition, this study explores how users adapt software to work with a larger system and thus create new options for product development under open innovation that point beyond improved product versions to improved systems.

Fourth, the findings may inform the economics of innovation for knowledge goods such as software. The open access to transaction-free zones such as FLOSS projects (Baldwin, 2008) may lead to cost advantages along the innovation process in terms of integration tasks because developers can access, reuse, and contribute to other projects' code base without the need for contracting, measuring, and billing. In addition, this study demonstrates how developers can save negotiation and communication costs by consolidating or setting up new (unencapsulated) transaction-free zones, within or outside existing project boundaries.

This study is one of the first to apply a perspective of sociomateriality to an empirical study (see Wagner et al., 2010). The difficulties encountered may be useful for future empirical studies to consider and improve upon. First, it proved difficult to adequately capture in words the observation of entanglement. Stumbling over formulations rooted in a logic of separatedness poses a challenge for research applying a sociomaterial perspective, both during the process of data gathering as well as during analysis and writing. This difficulty does not only apply to a researcher trained in rather modern views of science but also to the respondents in the field struggling to put into words their interdependent, co-emergent practices on multiple levels: friendships that happen to co-evolve with code, documentation written to help newcomers that in turn helps structure own work, saved development costs due to less negotiations while, nevertheless, constantly communicating, and many more.

Applying analytical separations to data that appear as entangled might be considered as incompatible with a sociomaterial perspective. The objection is valid and methodological approaches need to evolve in parallel with more refined empirical studies that employ a sociomaterial perspective. An economic analysis of organizational decisions builds on the evaluation of costs and benefits perceived by individuals and weighed against opportunity costs. It requires that individuals form expectations and make (boundedly) rational choices, in the case studied, for example, about organizational boundaries. Studying the economic rationale of organizational actors entangled in co-emergent and constitutive practices raises fascinating ontological and epistemological puzzles for research. Two small cautions can be derived from this study that might be worthwhile to pursue when refining empirical research strategies for a sociomaterial perspective: sociomateriality emphasizes the constitutive nature of entanglement that involves humans and nonhumans, such as software code. A consequence is that entities cannot be taken for granted independent of the performance of a practice (for a critical discussion see Faulkner and Runde, 2010). Particular attention to time may be warranted due to the changes in organizational contexts. Interviews with and observations of the same individuals and their actions at multiple points in time may inform researchers about

evolving practices and actors' evaluation of observable, economic activity in retrospect.

Second, a multi-level and multi-context approach to data gathering may prove valuable for studying the economic rationale of organizational actors. Sociomaterialty implies that technology is constitutive of the work in organizations including the meaning of specific actions as attributed by a collective of individuals and the identity of an individual engaged in a practice. These processes are highly relevant for economic decisions such as how to develop which kind of interface between technical components or who to approach for help. Orlikowski and Scott (2008: 465) suggest that: "if we can find a way to reveal the taken-for-granted, invisible dynamics of sociomateriality, it will enable us to generate deep insights into the contemporary world." Revealing such dynamics may require paying attention to the decision context at the cognitive, the collective, the technical, as well as the organizational (and here: inter-organizational) level because each level generates a context that will likely influence individual expectations and experiences that underlie economic decisions.

LIMITATIONS AND FUTURE RESEARCH

This study suffers from a few limitations. The choice of the Free Java environment as a sample was not a choice for the study of a particularly high or low rate of technological change, but for an environment that contained aspects of both (Garud and Kumaraswamy, 1995). The specifications for the Java language published by Sun Microsystems provided guidelines for the implementation of the technology. Beyond and within these specifications, FLOSS developers perceived ample freedoms for innovation and solutions that reached beyond the intended functionality.

A study of collaboration across a large sample of FLOSS projects could help to build a more general theory of cross-project collaboration. The study of a context where a given technology existed, at least in the form of specifications, was carried out in the full knowledge that it occupied a medium position, between radical and incremental, within the evolution of the technology (Utterback, 1994). Given the current context, collaboration and the observed social convergence were by no means inevitable. Historically, the projects were torn by license disagreements and personal feuds.

With respect to the social convergence in the sample, it is difficult to argue that this research represents a multiple case study. The observed convergence was a single process and possibly idiosyncratic for this sample (Lincoln and Guba, 1985). However, the notion of sociomaterial transaction rests on over 180 observations of technical collaboration across more than 30 projects. The non-obvious social convergence between the projects linked by individual networks that engage in technical collaboration invites further studies from a sociomaterial perspective. Environments with technologies that may work as integrated systems appear as particularly promising to study interwoven social and technical (inter-) organizational work practices.

The results reported challenge the established economics of systems integration and call for research on the organization of complex development projects. In contrast to prior work, which points to the necessity of a central systems integrator (person, team or firm), this study indicates that it is possible to distribute the tasks and authority for systems integration and negotiate user interfaces and the modes of interaction between different parts of the system. Furthermore, the problem of novelty – how to explore new ideas without compromising existing functionality – can be addressed by branching and/or friendly forking.

A number of questions remain unanswered. First. How common is distributed systems integration? Second, can these practices be observed in non-software contexts, for example, urban planning, education, or industrial products (Müller-Seitz and Reger, 2010)? Third, can distributed systems integration be applied within a firm, and what are the barriers to doing so? Fourth, can such practices be used across firms, for example to structure technical alliances or joint development projects? These questions alone make up a full agenda for future research.

More importantly, this paper has shown that distributed systems integration constitutes a viable method of coordinating relatively large FLOSS projects. Conceptually, this method of coordination is an alternative to "silo development" and centralized systems integration, which are regarded as "essential" features of proprietary development projects. When and why one method of coordination should be preferred to the other is a crucial question for future research in economics of innovation.

4. Social Practice

The development of complex innovations usually requires collective action because no single individual may likely embody the combined skills required. For the case of firms producing an output of appropriable, private goods coordinating collective action is well understood (Coase, 1937). Outside of firms, however, collective action is associated with struggles over the sharing of costs and benefits. The problem associated with collective action appears when considering the economics of public goods, which tend to be undersupplied or over-used, as in the tragedy of the commons (Olson, 1965; Hardin, 1968; Ostrom, 1999) due to the characteristics of public goods: non-rivalry and non-exclusiveness.

Studies of collective action, when the output is a public good, have focused on either non-routine events such as riots or revolutions or routine collective action such as rallies, protests, or a soccer game (Piven & Cloward 1992; Useem 1998). Knowledge is usually seen as a byproduct of collective action (Flora and Flora, 1993; Indyk and Rier, 1993; Myers, 1994) but rarely as its core product (von Hippel and von Krogh, 2003). With the collaborative development of technology, knowledge is the central output of (routine) collective action, open source software being its prime example (von Hippel and von Krogh, 2003).

Two literature streams in collective action are important in order to motivate a social practice perspective to gain theoretical weight going forward. Resource mobilization theory (see McCarthy and Zald, 1977 for a synthesis) contributed to an understanding of social movements as resting on organization to succeed. Rather than highlighting grievances and social unrest as crucial for collective action, resource mobilization holds that organization makes the difference between failure and success in mobilizing contributions and participation (Tilly, 1978). Second, process-based approaches to collective action focus on the dynamics in the work of collective action and the rewards available to contributors through participation (Elster, 1986). Elster writes:

> *"... cooperation reflects a transformation of individual psychology so as to include the feeling of solidarity, altruism, fairness and the like. Collective action ceases to become a prisoner's dilemma because members cease to regard participation as costly: It becomes a benefit in itself, over and above the public good it is intended to produce." (1986: 132)*

Collaborative innovation can be understood as a form of collective action where the central output is knowledge. Resource mobilization theory alerts us to the critical role that organization and coordination play for collaborative innovation. And process-based approaches remind us that identity and meaning change during the involvement in collective action (Melucci, 1996). This has economic consequences because it changes the individual perception of costs and benefits and can, thus, solve the free rider problem. It is this insight that the notion of private-collective innovation brings to organization science that dramatically improves our understanding of collective action when development processes are long-term, participatory, and knowledge-based (von Hippel and von Krogh, 2003; Spaeth et al., 2008).

However insightful the basic understanding of the terms of contributions is in economic terms, it simultaneously opens a wide set of questions about how collaborative innovation can be organized. Knowing that the free rider problem can be overcome when individual benefits outweigh the individual costs of contributing leads to the understanding that collaborative innovation can be sustainable (Roberts, Hann, and Slaughter, 2006; Hertel, Hermann, and Niedner, 2003). How this can be achieved is less clear, it is not obvious how organizations can implement processes and structures to support this form of collective action. The starting point is the process of collaboration because it is from the process that benefits emerge that have been missed in economic analysis that focused on the outcome only (exceptions are findings in behavioral economics that take into account, for example, warm-glow or reciprocity: Andreoni, 1990; Fehr and Schmitt, 1999).

This section speaks of social practice and it is all but a straightforward shift from the notion of studying a process to adopting a practice lens. In fact, this shift introduces an epistemological departure and an opening. The shift marks a break with methodological individualism and rational choice theory, two sets of assumptions underlying much of the work just cited in the last paragraphs. Attention to practice theory may yield valuable and complementary insights into processes of collaborative innovation despite non-compatible assumptions. When drawing on practice theory, I refer to accounts of relevance to social sciences more generally (Schatzki, Knorr Cetina, and von Savigny, 2001, Boltanski and Thévenot, 2006/1991) and to the study of technology in organizations more particularly (Leonardi, 2011; Feldman and Orlikowski, 2011).

Three reasons for attending to a practice lens stand out. They are, in a nutshell: experience, ethics, and material agency. First, the idea that a process yields benefits to individuals represents a cornerstone of private-collective innovation (von Hippel and von Krogh, 2003). Benefits that somehow accrue during and as part of a production process change and challenge the calculus of public good economics. Contributions no longer rest exclusively on the valuation of benefits connected to the availability of the final product, say, street lights. In public goods economics the willingness to pay depends on the benefits street lights provide. In aggregate, street lights are underprovided due to some individuals waiting for others with higher expected benefits to pay for the lights. Now, building the lights is no longer a black box and being part of the production process is beneficial, so far the analogy. And the example used is open source software development. Contributing to the development process of the public good that is open source software becomes worthwhile beyond the usefulness of the final product. Staying with the terminology of rational choice, we may account for the learning benefits involved when developing software jointly with peers (Spaeth et al., 2008), developers may experience fun and joy during work (Torvalds and Diamond, 2001), and they may see an opportunity to help others and interact in generative ways in the community (Lakhani and von Hippel, 2003). However, doing what developers do is rooted in causes that go beyond the rational and the three examples just cited stand in for wider developmental and personal motives best captured by the practice lens. Learning is also the path of becoming a skilled software developer, fun can represent an experience and mental state of flow, and helping is a form of empathy towards others. Simply rewording these three "intrinsic motivations" reveals the inadequacy of rational choice in capturing individual experience. The scare quotes refer to the first essay in this section where we complement self-determination theory with a social practice lens based on the work by Alasdair MacIntyre (1985). Beyond motivation, attention to individual experience may allow theorizing the acquisition and sharing of tacit knowledge and its application in skillful work, which plays a fundamental role in knowledge creation theory (see Niccolini, 2011). The concluding section five discusses aspects of individual experience for an agenda based on a practice lens.

Second, collaborative innovation is a collective activity spread over time. With a focus on technological innovation we know that the technology can enter as a significant way of life for the engineer (the developer) and shape her way of thinking and acting and experiencing the world (Martin 2002; Flyvbjerg, 2001, Raymond, 1999). In the case of open source software development, Raymond (1999) and Stallman (1999) were the first to point to the defining role that the way software was developed played in the life of the developer. The sharing of code and the responsibilities for technological progress and social, and communal development was seen as a defining aspect of being part of the Free and open source software movement (see also Himanen, 2001). Hence, the moral choice of being part of the movement or the community and contributing to its functioning and growth can be understood by considering the process of development rather than the outcome or final product. Reading Stallman and Raymond may even suggest that ethical consideration drive collaborative innovation in open source software. The first essay in this section takes a close look at the moral component of individual motivation to contribute and argues that self-determination theory may in fact be fruitfully complemented by a more long-term, developmental perspective on why individuals keep engaging in collaborative innovation. The key theoretical term is social practice. The contemporary moral philosopher Alasdair MacIntyre helps in drawing a wider circle around the moral consideration that software development practitioners such as Richard Stallmand advocate. A social practice, that can be collaborative innovation as practiced in a specific community of developers, works as a school of virtue (MacIntyre, 1981). This means that, by becoming a member of the social practice, the individual is socialized

into a way of doing things according to standards of excellence that both shape the individual sense of the right thing to do as well as evolve the community. The standards of excellence are worked out in daily activity during the process that is collaborative innovation. They are shared and refined in the community of practitioners. Hence, the practice over time, the process rather than the final product follows ethical considerations by developers who exercise judgment and responsibility as part of their engagement with a community and their (evolving) sense of belonging to that community.

Third, theorizing the emerging properties of technology in development is critical for understanding work contexts and strategic choice in a myriad of practical domains (Haefliger et al., 2011, see section one on strategy). A practice lens may help capturing the role technology plays in structuring and shaping organizational processes because it takes seriously the embodied and material context of collaborative innovation. A framework of rational choice may risk glossing over the material context by assuming a social component somewhere in the process and focusing on the outcome. Feldman and Orlikowski (2011) point out that organizational outcomes depend not on the specific technology or how they are generally used but on the enactment of technologies in practice as the technology is recurrently used and constituted in everyday action. While the authors refer to organizational outcomes of any type, collaborative innovation as the organizational process in focus here is affected directly. The decision what to develop next and how to expand and improve the technology under development depends on prior choices and practical contingencies (such as, for example, modularity, see Baldwin and Clark, 2006). These contingencies can be theorized as material agency which overlaps in constructive ways with human agency (Leonardi, 2011; 150). This thesis does not build further on this argument and I leave this angle on practice theory as simply a promising reason why to consider the practice lens in the study of collaborative innovation, pointing the reader to the last section of this thesis.

The papers in this section combine the study of communities of practice with theorizing on social practice as well as an observation of user behavior in a large online community.

CARROTS AND RAINBOWS:

MOTIVATION AND SOCIAL PRACTICE IN OPEN SOURCE SOFTWARE DEVELOPMENT

Georg von Krogh
Stefan Haefliger
Sebastian Spaeth
Martin Wallin

Chair of Strategic Management and Innovation
Department of Management, Technology, and Economics (MTEC)
ETH Zürich, Switzerland

Forthcoming in MIS Quarterly

Open source software (OSS) is a social and economic phenomenon that raises fundamental questions about the motivations of contributors to information systems development. Some developers are unpaid volunteers who seek to solve their own technical problems, while others create OSS as part of their employment contract. For the past ten years, a substantial amount of academic work has theorized about and empirically examined developer motivations. We review this work and suggest considering motivation in terms of the values of the social practice in which developers participate. Based on the social philosophy of Alasdair MacIntyre, we construct a theoretical framework that expands our assumptions about individual motivation to include the idea of a long-term, value-informed quest beyond short-term rewards. This "motivation-practice" framework depicts how the social practice and its supporting institutions mediate between individual motivation and outcome. The framework contains three theoretical conjectures that seek to explain how collectively elaborated standards of excellence prompt developers to produce high-quality software, change institutions, and sustain OSS development. From the framework we derive six concrete propositions and suggest a new research agenda on motivation in OSS.

INTRODUCTION

Over the last 20 years, open source software (OSS) products have made successful inroads into many information systems (IS) segments, attracting millions of users. OSS, broadly defined, is software where users can inspect the source code, modify it, and redistribute modified or unmodified versions for others to use. Today, firms are both heavy users of OSS products and contributors to their development. As a result, IS managers are increasingly dependent on development resources outside their direct control, giving them reason to be concerned about what motivates developers outside the firm, many as volunteers, to participate in the creation of OSS. For example, if a firm decides to invest millions of dollars to migrate its servers to a Linux system, managers will want to know to what extent Linux will continue to receive contributions from individuals and companies, how the software will evolve, and if the OSS projects involved will manage to release new and improved versions of the software regularly. Thus, IS managers should be interested in the question formulated by Lerner and Tirole (2002) which points to the very existence of the OSS phenomenon: "Why would thousands of top-notch software developers contribute for free to the creation of a public good?" A public good, here, is defined by its non-excludability and non-rivalry in consumption, which applies to software published and licensed under an open source license. Lerner and Tirole's question poses huge challenges for scholars studying IS development within firms that systematically rely on pay and career incentives. Answering this question explains a phenomenon of high academic and practical interest and much research on motivation in OSS has already shed considerable light on critical issues regarding contributions to OSS projects and the emergence and growth of OSS projects and their organization (e.g., Hertel et al. 2003, Markus, 2007; Ulhøi, 2004; West and O'Mahony, 2005).

In this paper we review the literature on motivation to contribute to the development of OSS. The review shows that the existing literature does not provide satisfactory answers to three more differentiated questions as to why this phenomenon exists (cf. Lerner and Tirole, 2002): First, why do OSS developers produce high-quality software when they do? This question is warranted because software quality is critical for attracting interest in OSS, as high-quality OSS systems make headlines due to their developers' achievements in terms of reliability, speed, accessibility, and more. For example, the world's largest stock exchanges run Linux systems while racing for trading speed records (Vaughn-Nichols, 2009; King, 2010) and 75% of the world's websites are served by OSS web servers. However, as our review will show, while the existing literature informs us well how developers are intrinsically and extrinsically motivated to contribute by means of time, effort, and code contributions, little is known about why they develop high-quality OSS when they do.

Second, why do OSS developers change institutions? A significant impact of the phenomenon on business and public administration stems from the nature of the free and OSS licenses that define and govern their potential use. Public administrators perceive value in the free accessibility of OSS (Comino et al., 2010; Maldonado, 2010) and firms have long used and contributed to OSS (Bonaccorsi et al., 2006; Dahlander and Wallin, 2006; Henkel, 2006; Stam, 2009; Rolandsson, 2011) as part of a model of innovation that has become known as private-collective (von Hippel and von Krogh, 2003). Thus, it is critical to explain why OSS developers set up OS licenses, organizations, and foundations and thus create new institutions that house what they do and make. However, while existing academic work explains how institutions constrain developers' motivation, it does not address more broadly how developers are motivated to change institutions.

Third, why do developers sustain OSS development? This question is critical because the answer indicates whether firms and individual users can expect future development of the software products in use today.[64] In this regard, volunteering OSS developers may constitute an unstable development resource as they can stop development or leave the project anytime. Developers may also suddenly and opportunistically alter their

[64] See The Economist, May 16, 2006 – "Open, but not as usual." http://www.economist.com/node/5624944?story_id=5624944

sharing behavior (Osterloh and Rota, 2007). However, so far, theory and research have not investigated in much detail why OSS developers engage in activities that guarantee the survival and sustainment of the OSS development practice.

In order to explore the three questions, we first review existing literature on motivation to contribute to OSS development and subsequently advance a new theoretical framework that deepens, enriches, and reinvigorates research on motivation in OSS. We show that the concepts of individual motivation and social practice are mutually constitutive, and argue that theory and research should incorporate a social practice perspective that focuses on action to explain how shared beliefs are created through shared work (Orlikowski, 2005, 2010; Rouse, 2007; MacIntyre, 1981), rather than reducing motivation studies to a conventional model that relies on short-term intrinsic and extrinsic motivation (e.g., Hars and Ou, 2002). A simple analogy illustrates our argument: Humans often act to satisfy their immediate needs and might fall for the dangling carrot. Occasionally, humans also make elaborate detours, strive for bigger things in life and undertake long voyages to find the gold at the end of the rainbow (even though they know it is not there). They value the journey, perceive unity in the sum of their efforts, and manage their life so that they can reflect upon it as well lived in a social practice. The new theoretical framework we put forward is based on the social philosophy of Alasdair MacIntyre. MacIntyre's perspective of social practice and motivation rests on a belief that human behavior cannot be decoupled from ethical considerations about what people strive for and the narratives they construct about their life. His work provides three important conceptual building blocks—goods, institutions, and social practice—that assist in the analysis of motivation to act. While much is known about the direct link between individual motivation and output in the form of, for example, submitting new or modified code to OSS projects, much less is known about how motivation is intertwined with forms of cooperative human activity (which we analyze through MacIntyre's concepts of "social practice" and "institution") and a broader set of related outputs, including high-quality technology for public use (which we analyze through his concept of "good").

With regard to motivation, OSS differs from traditional software development in firms along three dimensions: incentives, control, and coordination mechanisms. First, to motivate software development, firms rely on pay and career incentives, and other benefits stipulated as part of employment contracts (e.g., Peters, 2003). The presence of a significant number of volunteers in OSS development (Hars and Ou, 2002; Lakhani and Wolf, 2005) limits the effectiveness of incentives to firm employees; volunteers find motivation elsewhere. Second, and related to the use of incentives, software development firms implement control mechanisms, including behavior-, or output-based control (see Ouchi, 1978). The lack of formal governance and mandatory or formal membership by volunteers excludes managers from exerting control in a OSS project following these traditional ways (Markus, 2007). OSS is predominantly characterized by clan control, which is based on common values and beliefs (Ouchi, 1980), or self-control, based on self-monitoring (Kirsch, 1996; Ouchi, 1979). Third, firms tend to coordinate software development projects through organizational hierarchy and centralized planning (Cusumano and Selby, 1995). In contrast, coordinating the work of distributed OSS developers hinges on their ability to exchange information over the Internet at low cost (Bonaccorsi and Rossi, 2003). A set of rather simple tools, such as concurrent versioning systems and mailing lists, are used for coordination at the level of information exchange (Lee and Cole, 2003).

These three dimensions illustrate differences that can be expected between the social practice of OSS development and those normally associated with software development in firms and software engineering (Scacci, 2002). Moreover, the social practice of OSS development is saturated with ethical aspects. Historically, open source software grew out of the Free Software movement founded by Richard M. Stallman, which was primarily driven by ethical considerations; running software that a user cannot inspect, modify, and share is considered immoral. Developers who identify themselves with the Free, or Libre, Software movement are often motivated by the desire to allow users to control the software they require (Stewart and Gosain, 2006). Thus, motivation of OSS developers should also be studied from the perspective of the social practice in which the development work collectively unfolds, evolves, and heeds ethical considerations (Rouse, 2007). In the following, we will argue that MacIntyre's theory accomplishes precisely this task.

Next, we describe the method used for the review and examine existing theory and research, which predominately covers the link between motivation and contribution and, to a far lesser extent, the link between motivation, quality, institutions, and sustainable practice. Subsequently, we introduce and critique MacIntyre's theory, which is suited to further exploration of our three questions pertaining to motivation. In a fourth section, we develop a new motivation-practice framework that allows us to answer these three questions. The framework contains a set of theoretical conjectures from which we derive concrete propositions and suggest new research opportunities. The last section concludes the paper.

RESEARCH AND ON OPEN SOURCE AND MOTIVATION

Review Method

We performed a literature review of publications pertaining to motivation in OSS. The review aimed to (1) reveal in what way and to what degree literature has addressed the three questions we posed in the Introduction; (2) organize and classify received literature according to topics covered and their theoretical underpinnings, focusing on motivational aspects; and (3) identify gaps in the current literature that justify the creation of a new framework.

The review stages conducted correspond to those recommended by Cooper (1998: Problem formulation, data collection, data evaluation, analysis, and presentation. In doing so, we were able to take advantage of the suggestions and potential pitfalls associated with each stage of the review. The reviewed papers originated in different disciplines, including organization and management theory, anthropology, economics, law, psychology, and sociology, and applied a variety of qualitative and quantitative research designs. We therefore report the data and methods used for each paper reviewed (Table A1 in the Appendix).

We adopted a broad hierarchical search strategy to capture high-quality and relevant articles starting with the most reliable sources and adding not-previously-identified articles in subsequent searches. The search strategy followed four main steps (the first three were carried out on June 22, 2009, while the papers in the fourth category were added on a continuous basis). First, we identified all articles listed in the ISI Web of Knowledge database published by Thomson Reuters containing the terms "open source," "motivation," or "incentive(s)" in the title, abstract, or keywords. This index is the most critical source of published material since it contains published and peer-reviewed articles. The initial search yielded 214 articles; we excluded those that did not specifically deal with the relationship between motivation and OSS. Search results that described software that happened to be open source were also excluded, as were articles that focused exclusively on technical issues.

Our review focused on motivation in the practice of software development under open source licenses and excluded other activities carried out in electronic networks of practice (Wasko and Faraj, 2005), online communities (Wiertz and de Ruyter, 2007), and social networks sites more generally (Boyd and Ellison, 2007). While motivation patterns might be similar across various activities we have no indication to combine in one review motivation studies in different practices simply because they might occur online or share volunteer contributions. In total, we included 26 articles from the ISI Web of Knowledge database. Second, we browsed the paper repository at www.opensource.mit.edu to identify articles that matched the ISI search criteria. This online paper repository is a major source of unpublished papers and work in progress dealing with OSS. We included eight articles from this source.

Third, in order to be more comprehensive, we searched Google Scholar with the combined search terms "open source" and "motivation." Google Scholar searches the scholarly literature and identifies articles from multiple disciplines and sources: Peer-reviewed papers, theses, books, abstracts, and articles, from academic publishers, professional societies, preprint repositories, universities, and other scholarly organizations. The initial search yielded 38,900 (often duplicated) articles. Because of the vast number of articles identified we opted to retain only the top 200 articles. Google Scholar ranks articles "the way researchers do, weighing the full text of each article, the author, the publication in which the article appears, and how often the piece has been cited in other scholarly literature. The most relevant results will always appear on the first page."[65] In line with the two previous searches we excluded articles not specifically dealing with the relationship between motivation and OSS. We included one additional article that appeared only in Google Scholar. In addition, we searched Google Scholar using slightly different search phrases, such as "incentive" and "open source software." However, this search failed to yield any relevant articles to add to the full sample. Fourth, we included topical conference papers and book chapters that were known to us and colleagues in the field (and which had escaped the previous identification). This yielded six additional articles.

Finally, we only included English-language documents, which led to the exclusion of one ISI paper. The search strategy led to 40 articles in

[65] http://scholar.google.com/intl/en/scholar/about.html, retrieved June 22, 2009

our review sample, with a comprehensive inclusion of empirical contributions but a selective inclusion of purely theoretical contributions. The sample included all articles that empirically investigate the relationship between motivation and OSS development but excluded non-empirical, non-ISI articles that do not develop new theoretical categories or explanations. Conceptual papers were examined for motivational factors that were used to create theory. Motivational factors that proved relevant in empirical papers were also included in the review.

We coded all papers according to motivational aspects covered, as well as to how they regarded institutional context. For example, surveys proposing "I am contributing software because I learn from receiving feedback on my work" led to the creation of a "learning" dimension of individuals' motivations to contribute. If a study used a different terminology, but the specific motivation seemed sufficiently close to an existing one in the taxonomy, it was merged into the existing category. After a first round, we convened and performed a triage of all studies and motivational dimensions, merging them when they seemed to express the same type of motivation. This led to a final typology of ten clusters related to individual motivation. As a final step, we decided whether each cluster covered intrinsic motivation, internalized extrinsic motivation, or extrinsic motivation.

While reviewing the articles, we also categorized topics relating to institutions (which are outside of the direct control of a single individual), leading to a list of five dimensions, as discussed later. A complete list of papers included, with a brief description of methods and data used, is included in Table A1 in the Appendix.

The Motivation to Contribute

Studies of individual motivation to contribute to OSS development were grouped into ten motivational categories. We also examined the frameworks used to justify and categorize motivations. While a wide variety has been used, the most frequent framework by far has been the distinction between intrinsic and extrinsic motivation in self-determination theory (SDT) (see e.g., Deci and Ryan, 1985; Gagné and Deci, 2005). This distinction is based on different reasons that bring about human action. An action is extrinsically motivated when it is performed to obtain some separable outcome, whereas an action is intrinsically motivated when it is done for the inherent interest or joy of performing it (Deci and Ryan, 1985). A number of empirical studies have shown that OSS developers have both intrinsic and extrinsic motivations for contributing to its development (Hars and Ou, 2002; Lakhani and Wolf, 2005; Roberts, Hann, and Slaughter, 2006; Wu, Gerlach, and Young, 2007), and we summarize how some of the works make use of SDT. Following Lindenberg (2001) and Lakhani and Wolf (2005), Osterloh and Rota (2007) distinguished between enjoyment-based intrinsic motivation and obligation-based or community-based intrinsic motivation. Their paper provided a theoretical overview, whereas Lakhani and Wolf (2005) presented survey data showing that both types of intrinsic motivation, as well as extrinsic motivation, impacted on work effort. Wu et al. (2007) also used the SDT framework to explain the continued intention to contribute to OSS projects.

Most of the studies using SDT focused on intrinsic motivations. For instance, Hars and Ou (2002) suggested that intrinsically motivated developers spend more time and effort in open source projects than extrinsically motivated developers, but did not examine this empirically. Other empirical studies concentrated on intrinsic motivation rather than extrinsic motivation. For example, Lakhani and von Hippel (2003) linked feelings of competence and fun to willingness to help other developers. Authors have also viewed motivation in relation to reciprocity, for example: Giving software patches as "gifts" to the community (Bergquist and Ljungberg, 2001; Wu et al., 2007); reciprocal helping behavior (i.e. helping because of having been helped or expecting to be helped) (Lakhani and von Hippel, 2003); or status motivation (Roberts et al., 2006). This work stands in contrast to Lerner and Tirole's early explanation of contribution, purely based on extrinsic motivation (2002). Few of the works on intrinsic motivation looked at how institutional context shapes motivations (and incentivizes to create or change institutions).

While alternative frameworks have been proposed, they are often closely related to Deci and Ryan's original framework of extrinsic and intrinsic motivation. For example, Bonaccorsi et al. (2006) distinguished between economic, social, and technological motivation, building on a taxonomy proposed by Feller and Fitzgerald (2002). Economic motivation is similar to extrinsic motivation, while social motivation roughly corresponds to intrinsic motivation. The authors also

suggested a third type—"technological motivation"—that includes benefits from learning and working with a "bleeding-edge" technology.

Attempts toward a broader and integrative framework can be found in Hemetsberger (2004) and Hertel et al. (2003). Hemetsberger viewed motivation as "self-interest" and "others-orientation." Self-interest was further divided into task- and product-related motivation (corresponding to intrinsic motivation); others-orientation, including long-term utilitarian goals and social significance (corresponding to extrinsic motivation), was divided into internalized group goals and values, and socio-emotional relationships. Hertel et al. (2003) extended a model of voluntary action in social movements proposed by Klandermans (1997). The authors included collective, norm-oriented, reward, and identification motives, and combined these with the "VIST-model" (Hertel, 2002) that explains individual motivation in virtual teams.[66] These frameworks go beyond self-determination theory in specific aspects but do not answer any of the three questions we developed in the previous section.

Most of the work on motivation based on the intrinsic/extrinsic framework can be grouped into intrinsic motivation, internalized extrinsic motivation, or extrinsic motivation (see Table 14 for an overview of the resulting dimensions). Some motivations are by definition extrinsic but developers could internalize them, so that they are perceived as self-regulating behavior rather than external impositions (Deci and Ryan, 1987; Roberts et al., 2006). These internalized extrinsic motivations include reputation, reciprocity, learning, and own-use value. Pure extrinsic motivations include careers and pay. The specific dimensions with key empirical findings of the papers in our sample are summarized in Tables A2–A4 in the Appendix.

Interaction among motivation factors has been given less attention, with the notable exception of crowding theory (Frey and Jegen, 2001), which predicts mutual adjustments between intrinsic and extrinsic motivation; introducing pecuniary incentives may not always increase the supply of a public good (Frey and Jegen, 2001). Osterloh and Rota (2007) argued that extrinsic motivation might crowd out voluntary sharing of software and other knowledge. Several surveys have shown that around 40 percent of the contributions to OSS emanate from paid working time (Hars and Ou, 2002; Hertel et al., 2003; Lakhani and Wolf, 2005; Luthiger and Jungwirth, 2007). However, crowding out of intrinsic motivations cannot be detected in empirical studies so far. Lakhani and Wolf (2005) found that intrinsic and extrinsic motivations co-exist, and Roberts et al. (2006) detected no crowding out of intrinsic motivation by extrinsic motivation. Osterloh and Rota (2007) argued that this might be due to a balance between intrinsic and extrinsic motivations and the "pro-social intrinsic motivation of a sufficient number of participants to contribute [...enforcing the rules of cooperation]" (Osterloh and Rota, 2007, p. 196). Increased reputation and career concerns ("status motivation") even enhance developers' intrinsic motivations (Roberts et al., 2006), although extrinsic motivation does crowd out "own-use value" motivation (a form of internalized extrinsic motivation) (Roberts et al., 2006).

To conclude, while research on motivation in OSS generated a clear link between extrinsic and intrinsic motives and contributions, it did not relate individual motivation to the quality of the contributions made. This leaves our first question ("How and why do OSS developers produce high-quality software when they do?") unanswered.

Motivation, institutions, and social practice

OSS development is a form of collective action that shapes institutions, and in turn enables individuals to contribute (von Hippel and von Krogh, 2003). Decades of research into other forms of collective action, ranging from lobbying and preservation of natural resources, to money collection for a good cause, have shown that institutions and individual motivations are interrelated (e.g., Morris and Mueller, 1992). As a result, we believe that it is important to investigate both the individual level and the social context of development in order to understand individual behavior in a social practice. This section reports on our findings on the relations between motivation, institutions, and social practice in the literature reviewed.

The research reviewed here mostly investigates why developers are moved to contribute to OSS

[66] VIST stands for valence (the subjective evaluation of goals), instrumentality (the perceived importance of one's own contribution), self-efficacy (team members' perceived ability to perform the required activities for the team task), and trust (the expectation of reciprocity rather than exploitation).

development. However, it has disregarded potential external influences and interferences, and focuses instead on the underlying needs, wishes, and goal orientations of individuals. Early on, though, Deci and Ryan (1985) pointed to the link between a context (the external environment) and individual motivation. In self-determination theory, extrinsic and intrinsic motivations are both predictors and outcomes of institutional arrangements such as governance or norms, depending on "the nature of the study and the way and time in which self-determined motivation is measured" (Sheldon and Krieger, 2007, p. 885). Accordingly, some OSS scholars have recently moved beyond the direct link between motivation and contribution, and investigated factors that enable and constrain motivation and contribution to OSS. These factors are external to individuals and often beyond their direct control, impacting indirectly on OSS contributions. They do so either by influencing individual motivation or moderating its effect. While some authors discuss the interrelationship between the motivation of OSS developers and contextual factors that impact on development, most of this work is recent and difficult to categorize (e.g., von Hippel and von Krogh, 2003; Shah, 2006). Yet, our review was able to identify five kinds of contextual factors of relevance to our second question ("Why do OSS developers change institutions?") and the third ("Why do developers sustain OSS development?").

First, a few studies related motivation to institutions in terms of governance, community sponsorship, the provision of rewards, and license restriction. Markus (2007, p. 152) defined OSS governance as "the means of achieving the direction, control, and coordination of wholly or partially autonomous individuals and organizations on behalf of an OSS development project to which they jointly contribute." While governance in OSS has been described in terms of structure, practices, rules, and norms, it leaves unanswered an important question relating directly to motivation: What is the source of control in OSS development communities (Markus, 2007, p. 153)? As well as exerting control, the creation of routines and rules impacts on developers' voluntary engagement and motivation. Markus's question also points to the organizational sponsorship of OSS development. Shah (2006) distinguished between "open" and "gated" source communities, where "gated" refers to developers' limited access to the development process, due to firm sponsorship and control. She found that, in the long run, developers who are mainly motivated by use value tend to contribute to gated source communities, whereas developers mainly motivated by enjoyment contribute to open source communities. Stewart, Ammeter, and Maruping (2006) investigated the role of community sponsorship and distinguished between market (e.g., firm) and non-market (e.g., university) sponsors and concluded that developers pick up cues as to the project's future from the type of sponsor, impacting incentives to contribute. For example, they found that projects with a non-market sponsor attracted greater development activity than projects without a sponsor. How contributions are rewarded also seems to matter. Alexy and Leitner (2011), in a working paper, criticized existing OSS literature for assuming that intrinsic motivation and extrinsic financial rewards have a one-dimensionally positive effect on the motivation of individual developers; they claimed that payment norms moderate the effect on intrinsic and total motivation. It has also been found that license restriction impacts on developers' motivations, in the sense that less restrictive licenses tend to attract more development activity to a project (Fershtman and Gandal, 2007; Stewart et al., 2006). The literature reviewed thus far considers institutions as constraints to motivation and has studied the interaction and interference that governance, community sponsorship, provision of rewards, and license restrictions inflict upon individual motivation. That is, causality has been established between the way institutions impact on motivation and human behavior. However, the opposite causal direction has not been investigated although it is known that OSS developers set up governance structure (O'Mahony and Ferraro, 2007). This means that question 2 ("Why do OSS developers change institutions?") remains unanswered.

Second, a number of studies related motivation to norms and exposure to a specific community. The literature reviewed identifies social and technical exposure to a community that over time creates opportunities, in terms of advancement within the community's social structure (von Krogh et al., 2003; Rullani, 2007) and of insights that can lead to more challenging tasks (Shah, 2006). To get their work accepted by established developers and to be able to exert influence on the technical design in the project, newcomers have to adhere to behavioral scripts for joining the community. Learning about and following

"joining scripts" takes time for developers, but this time is essential if they are to advance to community leadership or other central positions (von Krogh et al., 2003). Although this literature approaches the idea of a social practice, question 3 ("Why do OSS developers sustain the social practice of OSS development?") remains unanswered.

Reviewing state-of-the-art literature, we identified five dimensions (governance, community sponsorship, provision of rewards, license restrictions, and social and technical exposure to the community) associated with institution and social practice that impact on motivation to contribute source code (see Table 15). However, little is yet known about what moves OSS developers to 1) produce high-quality work when they do, 2) engage in institutional change, or 3) sustain OSS development. In order to address these issues, we suggest research to take a social practice perspective on OSS development.

To summarize, individual motivation rooted in people's search for immediate outcomes is important but does not suffice to explain critical facets of the OSS phenomenon. The OSS phenomenon initially triggered research on contributors' motivation from a self-determination perspective, but to get more differentiated explanations we may need to look beyond this dominating perspective. Some of the work reviewed represents a first step toward taking into account institutions and practice; yet it seems that the emergence of institutions through social practice, and their interactions in OSS development, are currently not well understood. A perspective that explicitly engages with these interactions can be found in the social philosophy of Alastair MacIntyre, which we introduce in the following section. Subsequently, we develop a framework that contains theoretical conjectures from which we derive concrete propositions. The framework fills the identified research gap and thus answers the three questions posed in the Introduction.

	Intrinsic				**Internalized Extrinsic**				**Extrinsic**	
	Ideology	**Altruism**	**Kinship**	**Fun**	**Reputation**	**Reciprocity**	**Learning**	**Ownuse**	**Career**	**Pay**
Alexy and Leitner, 2007										✓
Baldwin and Clark, 2006										
Benkler 2002				✓						✓
Berquist and Ljungberg 2001						✓				
Bitzer et al. 2007		✓						✓		
David and Shapiro, 2008	✓		✓			✓	✓	✓		
David et al. 2003	✓					✓	✓	✓		
Fershtman and Gandal, 2004										
Ghosh 2005	✓	✓			✓		✓	✓	✓	✓
Hars and Ou 2002		✓	✓		✓		✓	✓	✓	
Haruvy et al. 2003		✓								
Hemetsberger 2004	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Hertel et al. 2003	✓		✓	✓	✓			✓	✓	✓
Ke and Zhang 2008		✓								
Lakhani and von Hippel 2003	✓			✓	✓	✓		✓		
Lakhani and Wolf 2005	✓		✓	✓	✓	✓	✓	✓	✓	✓
Lattemann and Stieglitz 2005					✓			✓		✓
Lee and Cole, 2003										
Lerner and Tirole 2002					✓				✓	
Luthiger and Jungwirth 2007				✓						✓
Markus, 2007										

O'Mahony and Ferraro, 2007										
Okoli and Oh 2007					✓					
Oreg and Nov 2008		✓			✓		✓			
Osterloh and Rota 2007		✓						✓		
Riehle 2007									✓	
Roberts et al. 2006				✓	✓		✓	✓	✓	✓
Rullani, 2007										
Schofield and Cooper 2006		✓	✓					✓		
Shah 2006	✓			✓		✓	✓	✓		
Spaeth et al. 2008					✓		✓			
Stewart et al., 2006										
Stewart and Gosain 2006	✓	✓			✓		✓			
von Hippel and von Krogh 2003				✓			✓	✓		
von Krogh et al., 2003										
Wu et al. 2007		✓					✓	✓	✓	
Xu et al. 2009	✓			✓	✓		✓	✓		
Ye and Kishida 2003							✓			
Yu et al. 2007	✓	✓			✓	✓	✓		✓	
Zeitlyn 2003			✓							

Table 17: Intrinsic and extrinsic motivations

Governance	Governance refers to "the means of achieving the direction, control, and coordination of wholly or partially autonomous individuals and organizations on behalf of an OSS development project to which they jointly contribute" (Markus, 2007: 152). *Key empirical findings:* • Benkler (2002) argued that peer production has a relative advantage over firm- or market-based production because of the highly variable nature of human capital and lower costs of coordination and communication. • Lattemann and Stieglitz (2005) argued that the adequacy of governance tools is related to the motivational preferences of participants and that behavioral and output control should be regarded as secondary to social control in the form of morals and cultural rules. • Lerner and Tirole (2002) argued that the success of an OSS project is dependent on its ability to break it into distinct components as well as the presence of a credible leader or leadership. They suggest that code modularity enables the former and exemplifies the latter with projects characterized by authority or consensus. • Markus (2007) suggested three purposes of OSS governance: Solving collective action problems, solving coordination problems, and creating a better climate for contributors. Her review also suggested a framework for future comparative and case study research on OSS governance. • O'Mahony and Ferraro (2007) concluded that, contrary to common belief, the contributions the community valued the most were not purely technical. The study showed that over time a governance structure emerged that valued the informal work of coordinating individual efforts and linking them to community goals. In doing so, the members developed a shared basis of formal authority but limited it with democratic mechanisms. • Shah (2006) distinguished between open and gated software communities, performing a qualitative study of both, and found that the difference in governance structure affects the participation choices of volunteer OSS developers. In addition, she found that in the long run, developers who were mainly motivated by use-value

	tended to contribute to gated source communities, whereas developers mainly motivated by enjoyment contributed to open source communities. Considerations of fairness and reciprocity explain this behavior. The developers were aware of the property rights situation in the gated source communities and suspected firms of acting "strategically," that is, neglecting the needs of the community and pursuing their economic interests. Thus, developers in gated source communities contributed only if they derived direct use-value from their work.
Community sponsorship	Sponsorship refers to control by an organization, such as a firm, over the development process, source code accessibility, or code ownership. *Key empirical findings:* • Shah (2006) studied open and gated software communities and argued that firms may encounter difficulties if they seek to construct hybrid arrangements that balance community-based value creation with private value appropriation. • Stewart et al. (2006) found that community sponsorship and licensing address complementary developer motivations, so that the influence of licensing on development activity depends on what kind of organizational sponsor a project has. In addition, OSS projects with sponsors—non-market sponsors in particular—attract more attention.
Provision of rewards	The provision of rewards refers to incentive structure, i.e., how someone other than the developer is rewarded for participation in OSS development. *Key empirical findings*: • Alexy and Leitner (2011) found that payment norms moderate the effect on motivation. In particular they studied the impact of rewards and payment norms within a community on developers' intentions to contribute to an OSS project. They found that payment exhibited a positive effect on developers' total motivation when they were offered a monetary reward on completion as long as payment was not strongly expected. At the same time intrinsic motivation only decreased for that share of the sample that received payment, which also perceived the existence of social norms refuting payment. • Benkler (2002) distinguished between, and related, monetary rewards, intrinsic hedonic rewards, and social-psychological rewards (such as social associations and status perception). He formalized the possible effects of increased monetary rewards. In particular, he discussed the situation when one agent is jealous of another's rewards (which he called *jalt*), such as when some people are paid and others not. • Lerner and Tirole (2002) drew on labor economics when they analyzed the economics of OSS and concluded that OSS contributors could be directly rewarded though employment or rewarded by signaling and subsequently gaining employment. • Roberts et al. (2006) found that paid participation lead to above-average contribution levels. In addition, they found that being paid to contribute was positively related to developers' status motivations but negatively related to their use-value motivations. No evidence of diminished intrinsic motivation in the presence of extrinsic motivations was found.
License restriction	License restriction is usually thought to limit the commercial exploitation of the source code, i.e., the option to combine OSS with proprietary software for sale. *Key empirical findings*: • Fershtman and Gandal (2007) found that output per contributor is much higher when licenses are less restrictive and that the number of contributors is higher when licenses are restrictive. • Lerner and Tirole (2002) described changes toward less restrictive OSS licenses that opened up for bundling OSS with proprietary software. They discussed the firm advantages of releasing existing proprietary code. • Osterloh and Rota (2007) claimed that OSS can flourish even after the advent

	of a dominant design, when knowledge sharing is no longer supported by great learning potential, low opportunity costs, and selective benefits. They argued that this is because OSS is better at solving first- and second-order social dilemmas where OSS licenses hinder the exploitation of voluntary donors. • Stewart et al. (2006) found that community sponsorship and licensing address complementary developer motivations, so that the influence of licensing on development activity depends on the kind of organizational sponsor a project has. For example, "the presence of a non-market sponsor may alleviate concerns as to the project's future in the same way as a restrictive license would, in the sense that the restrictive license is not perceived as necessary to protect the developers' interests" (2006: 141). In addition, OSS projects that use a nonrestrictive license will attract greater user interest over time than those that use a restrictive license.
Social and technical exposure to the community	Refers to effects from exposure that, over time, create opportunities for advancement and work on more challenging tasks that require deep insights into the community's workings. *Key empirical findings:* • Rullani (2007) concluded that exposure to a community increases developers' contributions independently of their pre-determined preferences. • Shah (2006) maintained that the developers' long-term accumulation of knowledge provided opportunities to engage in more challenging tasks, or tasks that require broader knowledge of the code base (multiple modules and components). • Spaeth et al. (2008) found that contributions impacted on developers' positioning in the community, which in turn provided some developers with private benefits, such as reputation gain, influence of the technical agenda, and learning opportunities. • von Krogh et al. (2003) found that newcomers who wanted to get their work accepted by established developers and to be able to exert influence on the technical design in the project, have to adhere to behavioral scripts for joining the community.

Table 18: Institutions and social practice

OSS DEVELOPMENT: CONCEPTUAL BUILDING BLOCKS

In recent years, scholars have increasingly shown interest in the influence of social practices on the evolution of economy and society, often referred to as the "practice turn" in the social sciences (Schatzki, Knorr-Cetina, and Savigny, 2001). The practice turn has also influenced theorizing in IS development and use (e.g., Kellogg, Orlikowski, and Yates, 2006; Orlikowski, 2000; Suchman, Blomberg, Orr, and Trigg, 1999) and stimulated theory development on human motivation that takes into account the relationship of passionate individuals engaging in their social practices and the role individuals play in the institutionalizing power of social practices (Brown and Duguid, 1991). The practice turn describes the work of social scientists who attempt to articulate relationships between individual activities, structures, and matter (for example, rules or physical bodies) that have been separated by various dualisms (such as structuralism or methodological individualism). Alasdair MacIntyre's seminal work After Virtue (1981) presents a theory that extends and links aspects from the social practice perspective. The theory includes an ethical dimension to the discussion of quality in the goods produced by motivated individuals in a social practice that is either enabled or constrained by institutions (Knight, 1998).

MacIntyre's work has had wide-ranging implications for several fields, including sociology, political science, education, religious studies, and business ethics (for a discussion and introduction, see Knight, 1998). His perspective on human motivation is rooted in moral philosophy, in particular virtue ethics, which concerns the knowledge of how to live a good life (MacIntyre, 1981). As we showed in the last section, motivation is often considered a physiological feature aroused in human beings that make us act in the direction of a specific goal. However, according to MacIntyre, understanding what people do (or do not do) and why they do it cannot be decoupled from an analysis of ethical considerations, such as what they strive for in life, the narratives they construct about their life, and why these are worthwhile to them and others. The theory builds on, and criticizes, Aristotelean value ethics by arguing from a more relativistic and process-based point of view. Furthermore, he takes a dynamic and historical perspective that is rare in mainstream motivation studies and explores the role of institutions and social practices in shaping virtues.

More specifically for our purpose, MacIntyre's (1981) theory is useful for analyzing motivation in the context of OSS development because it raises questions of ethics that guide the activities of people, institutions, and social practices. As shown by Stewart and Gosain (2006), OSS developers often report that their work is guided by strong ideologies (which may be expressions of ethics), such as writing software that should be free for all to download and use. A review and framework of motivations in the context of the OSS development phenomenon therefore needs a theoretical foundation that allows such ethical and ideological issues to be addressed.

Further, MacIntyre (1981) is concerned with the implications of so-called "incommensurable moral premises" rather than approaches to reach a consensus on values, norms, and decisions often found in conventional (analytical) moral philosophy (e.g., Habermas, 1988, on rational dialogue and communicative action; see Mingers and Walsham, 2010, for an application to IS). The theory is based on the assumption that people from different cultures, practices, and institutions may "subscribe" to various ethics and thus may also argue from different moral premises that would be impractical or impossible to reconcile. Incommensurable moral premises are relevant for understanding motivation in OSS development, too. While commercial firms provide pecuniary and career incentives to motivate developers, many (but not all) OSS developers work for free (Lakhani and Wolf, 2005). According to field accounts, some developers even find it "immoral" to work for a commercial software company (Stallman, 1999).

Trying to change developers' moral premises may be difficult and impractical. Firm participation in OSS development and gated communities (Shah, 2006; Roberts et al., 2006) raises the question of implications for developers' motivation, given the co-existence and interaction of institutions and social practices that build on different moral premises. People who join a social practice gradually adopt shared values and strive toward standards of excellence defined within it (MacIntyre, 1981). Thus, the analysis of moral premises in a social practice is essential in order to understand what motivates people to perform and improve a craft. As Feller and Fitzgerald (2002) argue, software development is a craft that has

evolved its own professional standards, which constitute the context in which software development is done. It is reasonable to assume that in contemporary software development people are differently motivated and conduct their work based on vastly different (perhaps incommensurable) moral premises. An analysis of motivation in this context thus needs a framework that can capture these moral premises.

There are three concepts from MacIntyre's theory that are central to our analysis: Social practice, goods, and institutions. Human activity is a holistic expression of the narrative of a human life embedded in some social traditions that give rise to social practices and the pursuit of quality. MacIntyre defines a (social) practice as "any coherent and complex form of socially established cooperative human activity through which goods internal to that form of activity are realized in the course of trying to achieve those standards of excellence which are appropriate to, and partly definitive of, that form of activity, with the result that human powers to achieve excellence, and human conceptions of the ends and goods involved, are systematically extended" (MacIntyre, 1981, p. 187). This definition can be applied to a range of professions and crafts, such as architecture, medicine, journalism, science, and the arts, with the precondition that a social practice should have wide and positive effects for humankind.[67]

Because of its ubiquitous presence and the wide-ranging impact of IS in all aspects of contemporary human life, software development, like many other areas of engineering and technology (van der Burg and van Gorp, 2005; Latour, 1996), should be considered a social practice in MacIntyre's sense of the concept. Scacchi (2002), though, has written about the differences between traditional software engineering and OSS development practices (see also Rolandsson et al., 2011). He notes that "[OSS developers] enact teamwork structures and relatively flat, peer-oriented, decentralized community forms that reduce/supplant functional organizational forms inherent in traditional [software engineering] techniques that increased bureaucratic tendencies. [OSS] avoids reliance on formal project management techniques and administrative structures that pervade industrial [software engineering] projects" (Scacchi, 2002, p. 3). Moreover, the social practice is intertwined with the technical object (see also discussions in Orlikowski and Scott, 2008). MacKenzie (2005) suggests that the OSS code itself, with modular, functional, and transparent objects, gives rise to a social practice with its own ethics. The technical object of the software code requires developers to behave in a specific way when creating and maintaining it, for example, modularizing, reusing, keeping to the API specification, or taking great care to document (Baldwin and Clark, 2006). Thus, "to be an OSS developer" means to engage in a social practice and to adhere to its specific rules of conduct, because these are believed to enable the creation of a high-quality product for its users (von Hippel, 2001; von Krogh and von Hippel, 2006). At the same time, according to Scacchi, it also means abandoning other rules related to institutions of software engineering prevalent in industry

When individuals act, MacIntyre's theory proposes that social practices together with institutions create two types of good, external and internal. External goods (external to the social practice) include capital, status, or power, which are the property of individuals and/or institutions. Internal goods are defined by the social practice and are public goods that benefit all participants in the social practice and the wider community. This is why, as noted above, MacIntyre (1981) chooses to speak of a social practice in relation to collective activity that produces wide-ranging positive effects for humankind. The difference between external and internal goods can be exemplified in the case of science. Scientific knowledge can be considered an internal good of science, of benefit to the scientific community and humanity at large. The creation of scientific knowledge adheres to the highest methodological standards set by the scientific community. The status and salary bestowed on individual scien-

[67] A social practice is the basis for making decisions about which virtues are called for in particular circumstances and the best way of enacting them (Noel, 1999; Fowers, 2003; Rämö, 2004). Dunne (1992) refers to "ethical knowledge" that directs "ethical action." Aristotle's discussion of the "good" refers to man's practice "of his soul's faculties in conformity with excellence or virtue" (Aristotle, 2001: 33). The good becomes a metaphysical goal, like truth, justice, and beauty, toward which people strive by adjusting their life and actions. However, in a modern world "good" may be contested by people who pursue different goals. The very standards that define what is good may be subject to different interests and, therefore, judgment itself will be judged as more or less virtuous (whether something is judged good or bad, right or wrong). MacIntyre argues we should rather understand the common good as internal to a social practice, as a goal to be achieved by its practitioners (Knight, 1998).

tists, and the power that follows from their expertise, are the external goods of science. However, there is no short cut to obtain these external rewards, which illustrates the importance of practice-based virtues of practitioners extending beyond their personal preferences. One first needs to work hard on developing the skills necessary for being a good scientist (as an unpaid student, for example) in the view of the scientific community, before personal economic benefits can be realized. Analogously, internal and external goods exist in OSS development. Internal goods are, for instance, the resulting software code, which—under an appropriate license—has public good characteristics. Additional internal goods to the practice are the joy of collaborating with similar-minded peers and the spreading of quality software, empowering software users. Given the wide range of developers' skills, the code quality nevertheless varies across projects. Individual reputation, the opportunity for developers to signal their value to potential employers through their code, and the solution to one's private technical problems are external goods that are accessible by one individual.

MacIntyre's theory emanates from a criticism of Aristotle's work on ethics and virtues in political leadership. He notes that practitioners achieve excellence of character or virtue in pursuing internal goods. To act in a virtuous manner is to emulate the rules of morality rather than simply abiding by them because one is in one way or another commanded to do so. A social practice, therefore, is a "school of virtue," where practitioners learn aspects of the internal good, such as ethical reasoning, argumentation, criteria for excellence and product quality, rules of communication, and so forth. Justice, courage, truthfulness, and above all love for the social practice, are cultivated through practitioners' participation. Practitioners discover and commit to goals that lie beyond their own selfish, short-term needs and desires. They realize that they can only achieve the internal goods that are of value to themselves, their social practice, and wider society when they emulate the standards of excellence already established within the practice. Pursuing internal goods (excellence) is synonymous with cultivating virtues by subordinating oneself and one's relations with others to the reasoning that is internal to the social practice.[68]

As Feldman and Orlikowski (2011, p. 11) argue, "in focusing on practice theory, we understand the mutually constitutive ways in which agency is shaped by but also produces, reinforces, and changes its structural conditions." We refer to these structural conditions, which include organizations (e.g. hierarchy, community), rules (e.g. coordination), and routines, as institutions. More specifically, in MacIntyre's theory institutions are "characteristically and necessarily concerned with [the provision of] external goods" (1981, p. 194), and can be understood as sustainable forms of human cooperation, governed by these organizations, rules, routines, which exist beyond the presence and efforts of each individual. Thus, medicine, chess, and software development are examples of social practices, whereas a hospital, chess club, or software firm are institutions.

Institutions such as governments, NGOs, and firms are a prerequisite for the organization and sustenance of social practices. In the words of Beadle and Moore (2006), "institutions house the social practice." A social practice cannot exist and be sustained without the supporting structures that provide rules for human cooperation. The relationship between institutions and social practices is so intimate that they "form a single causal order in which the ideals and the creativity of the practice are always vulnerable to the acquisitiveness of the institution, in which the cooperative care for common goods of the practice is always vulnerable to the competitiveness of the institution" (MacIntyre, 1981, p. 194). At the same time as institutions enable social practices to produce internal goods, they distribute external goods in the form of power, status, and financial rewards. Thus, internal goods and external goods are always produced concurrently, giving rise to unavoidable tensions between social practices

[68] For there to be an identifiable common and internal good, MacIntyre (1994, p. 35) suggests: "…there must be identifiable structures of community, so that one can understand how the parts which different individuals contribute are contributing to a common goal." However, as Blackledge (2009, p. 870) argues in his analysis of MacIntyre and social practices: "It is … only in small-scale communities that politics can escape from the compartmentalization that is endemic in the modern world." MacIntyre believes that it is this compartmentalization that hinders the "flourishing of local communities" (MacIntyre, 1998a, p. 248) and that it is only in local communities that "cooperation as a common good" (MacIntyre, 1999, p. 114) can emerge spontaneously. However, the Internet allows even large global communities with hundreds of developers and users to follow each other's work, making Blackledge's insistence on "local" less relevant for analyzing OSS development.

and institutions to be discussed below. To summarize, the conceptual building blocks discussed here form a dynamic and mutually dependent complex that helps us to understand aspects of human behavior in general and cooperative behavior, such as OSS development, in particular.

MacIntyre's Critique of Firms and Managers

MacIntyre's theory has been used in organization studies (e.g., Beadle and Moore, 2006) but, as Dawson and Bartholomew (2003) note, applying it is not unproblematic since MacIntyre criticizes heavily contemporary notions of management and firms as institutions. While this concern must be taken seriously, we will show that the strength of the conceptual building blocks is precisely that they derive explanatory power from highlighting distinct differences between social practices and institutions.

Whereas institutions provide practitioners with external goods that satisfy their need for compensation, they may also corrupt or fail to support social practices by overshadowing or conflicting with motives to develop a craft to adhere to the social practice's ethics. Inevitably, much of MacIntyre's critique of institutions is directed at management (MacIntyre, 1981). However, Moore (2008) points out that this critique is part of a general critique of modernity; we live in an emotivist culture in which moral choices and actions are expressions of people's preferences and emotional states. In MacIntyre's highly controversial view, managers' preferences are to seek external goods as ends, using other individuals as means. They justify their power and monetary compensation by implementing techniques and systems for social change. However, because of the complexity of organizations, techniques and systems seldom lead to predictable outcomes in terms of enhancing effectiveness or efficiency, and so he claims the basis for justification must be false. Furthermore, the implementation of planned social change can do more harm than good to a social practice's capacity for producing internal goods. This critique echoes Robey and Markus's (1984) much-cited analysis of the unpredictable outcomes of the design of management information systems (MIS). Moreover, because managers are often influenced by monetary rewards and seek to create and appropriate excessive external goods, their actions may undermine practitioners' motivations to improve their social practice, and ultimately destroy its social fabric, including relationships between practitioners.[69]

Analyzing the "morality of management," Moore (2008) attempts to find a solution to this conundrum. In his view, the business organization is a social practice-institution combination where the presence of virtuous agents at both levels may result in a "virtuous business organization." The manager partakes in both the "core" of social practice of , for example, software development and the social practice geared at building and changing institutions (e.g., running a software firm). MacIntyre repeatedly warns that because institutional goals may conflict with the internal goods of a social practice, institutions may constrain or corrupt social practices and demotivate or demoralize practitioners. This is the case when institutions pursue limited goals aimed at external goods (e.g., excessive profits and competition) at the expense of internal goods that motivate practitioners toward the goals of improving the social practice, such as achieving excellence in a craft (making money on watches is not always the same as perfecting the skills of watchmaking). One strength of MacIntyre's work is to highlight such distinct differences between social practices and institutions. The role of virtue ethics also becomes clear: Without virtues such as justice, care, courage, and truthfulness, social practices could not resist what he calls the "corrupting power" of institutions (MacIntyre, 1981, p. 194).

Critique and Exegesis

We believe there are great merits to MacIntyre's theory when analyzing human motivation in relation to social practices and institutions. Yet,

[69] MacIntyre (1998b) also raises doubts that management can be considered a social practice because this presumes attention to the creation of internal goods and the well-being of humanity. Managers' concerns are with techniques and systems for transforming raw materials into products, unskilled labor into skilled labor, and investment into profit (p. 30). Managers decide and act to achieve a desired end state through social change, but are blind to other concerns, such as the wider effect of their actions on humanity. Thus, MacIntyre concludes they are not able to engage in moral debates about their own actions. MacIntyre's harsh critique is unwarranted. It neglects the very fact that managers are mostly concerned with the consequences of their actions and repeatedly engage in a wider discourse with society about the nature of their activities and the purpose of the institutions they run. A similar point is made by Dawson and Bartholomew (2003).

his writings have received fundamental criticism, which we will discuss briefly here. The following section thus completes our review of MacIntyre's theory with a critique and exegesis. This critical interpretation aims to uncover the significance of the theory for OSS development, and in particular how it informs the three questions we posed in the Introduction.

First, MacIntyre has been criticized for his claim that habitual patterns of behavior associated with ethical values can no longer be found in contemporary society in the way that Aristotle conceived of them in his idea of "polis" (Bender, 1998). MacIntyre gives too much credit to an historical account of virtues, and how they can be achieved by individuals seeking to obtain "unity of life." In so doing, he takes an overly optimistic view of what social practices, and the people within them, can achieve in terms of striving for excellence. Yet writers on OSS remind us that there are strong ethical values at work in the community of developers (Stallman, 1999; Zeitlyn, 2003). For example, people who are suspected of unlawfully commercializing the software, or of introducing code from a commercial environment, are likely to be "flamed" through verbal attacks and have their privileges (such as access to the formal version of the source code) revoked, because this behavior is not considered consistent with being an OSS developer (Shah, 2006). MacIntyre's theory links human behavior and standards of excellence within the social practice to outputs in the form of internal goods. One main concern for the software industry in general, and OSS in particular, is the output of high-quality software (see e.g., Aberdour, 2007; Gillies, 1992; Kan, 2002), which is captured by question 1: "Why do OSS developers produce high-quality software when they do?"

Second, Macintyre speaks broadly and confidently about managers and firms without seeming to care much about the details of what they do and how they work. While he takes a negative view of institutions (e.g., firms), at the same time he acknowledges "no practices could survive for any length of time unsustained by institutions" (MacIntyre, 1981, p. 194). MacIntyre's critique of institutions, in particular firms, is rooted in an overly critical view of modern society. Claiming that firms universally demoralize practitioners, who otherwise seek to do good for their craft and social practice, is naïve and does not take into account the variety of motivations and institutions in existence (see also Brewer, 1997). Instead, it is more fruitful to point to the risk that institutions may destroy social practices. For MacIntyre, managers act on behalf of institutions and are primarily concerned with increasing external goods. Yet, the pursuit of external goods clearly does not preclude some kind of balance between internal and external goods (Halliday and Johnsson, 2010). In our reading, MacIntyre is overly pessimistic about management's ability to achieve this balance. As we mentioned above, one solution to the problem is to understand managers as belonging to a type of social practice geared at founding and changing institutions (Moore, 2008). However, accepting this perspective risks omitting a potentially important inquiry into why OSS developers (rather than managers) contribute to institutional change. What is clear from MacIntyre's analysis is that institutions and social practices are mutually dependent for the production of goods, and it follows that changes in institutions may trigger changes in social practices and vice versa. The mechanisms by which this change functions, and the role of OSS developers, remain unknown but the link between institutions and social practice provides a theoretical basis for approaching question 2: "Why do OSS developers change institutions?"

Third, MacIntyre's view of virtue ethics as related to social practices and, more broadly, culture can be criticized from a utilitarian view of moral philosophy. Proponents of the latter view would seek to find universal rules that can guide moral behavior (Adams, 1976; Cornman, Lehrer, and Pappas, 1982; Mill, Bentham, and Ryan, 1987). For example, one person will help another in distress because helping is a universal rule that provides utility to the person being helped and brings the helper a deep sense of satisfaction. MacInytre's relativistic position neglects the search for such universal rules. In addition, from a utilitarian perspective, the theory defines an ideal unity of life bound to the social practice and the virtues it develops, but lacks objective standards. However, it would also be problematic to ignore entirely that moral premise and actions in everyday life may be shaped by the social context. Indeed, OSS development is a collective undertaking, whether in terms of reuse of software code or adhering to collectively established rules of conduct. Thus, researchers may acknowledge the social practice as the relevant context for OSS development and study rules that might depart from universal rules (say, of the Free Software movement or commercial software devel-

opment). In studying OSS development as a heterogeneous phenomenon[70] that contrasts with commercial software development institutions (such as software firms and intellectual property rules), MacIntyre's theory offers a way to think about motivation with a complementary set of assumptions about how and why people act in a particular context (see Section 4). This is particularly relevant because of the long-term perspective on the end result of making a contribution to OSS and the unclear rewards associated with engaging in other practice-enhancing activities (see Shah, 2006). The shift in the conception of motivation as primarily reward-oriented (as e.g., in self-determination theory) to practice-oriented may guide research into answering question 3: "Why do developers sustain OSS development?"

[70] There can be major differences, factions, and guilds among OSS developers. As one reviewer helpfully pointed out, there is an "XML guild," for example. Developers often campaign for the "right and best" text editor, programming language, or software license may be.

TOWARD A NEW RESEARCH FRAMEWORK AND AGENDA

The research framework and agenda that we present in this section grows out of the incomplete match between the nature of OSS development and the theoretical underpinnings of reviewed literature on motivation to contribute to OSS development. OSS development is characterized by significant voluntary contributions, self-selection of tasks among developers, clan- and self-control, community-type organization, and strong ethical considerations. Our review identified the dominant role played by self-determination theory in explaining developers' contributions. However, as we discovered, self-determination theory cannot account for several of the intricacies that characterize the OSS development phenomenon. As a result, the domination of the self-determination perspective that runs through the existing motivation literature may compromise our field's ability to adequately address the three research questions posed initially. There is no doubt that self-determination theory does provide very valuable guidance and answers to many important research questions. However, a fundamentally different and complementary approach is required to account for gaps we identified; research on motivation in OSS development neglected important aspects of the social practice as a context for motivation, including the ethics and virtues that guide the work of engineers and software developers (Friedman and Kahn, 1994; Latour, 1999; Martin, 2000, 2002). In short, people are moved to do many things, but whether or not this is constituted by and for the social practice of OSS development, is an entirely different matter.

In this section, we develop a theoretical framework to explain how virtues and ethics—integral elements of a social practice—influence developers' motivations to contribute to OSS. Grounded in the conceptual building blocks discussed in the previous section, we suggest an alternative and complementary set of assumptions about the individual OSS developer, as outlined in Table 16. In particular we emphasize a logic of unity of life. Following this logic, individuals do not necessarily act to achieve some immediate reward, because they want to maximize use-value or gain favors (compare this with the self-determination view). Instead, they act to achieve unity of life; they want to reach or maintain consistency of action throughout their life—an ambition that values personal development and contextual events and points beyond the attainment of specific and immediate rewards. We use the metaphor of the journey to the end of the rainbow to describe the idea of unity of life because a quest harbors value in itself and can be questioned, interpreted, and reconstructed self-referentially. The journey might be considered worthwhile, exhausting, beautiful, pointless, fulfilling, and many other things—but as a journey it follows a sequence of events limited by the individual life experience and subject to reflection, memory, and expectation. In MacIntyre's terms, human lives possess narrative structures because they are embedded in social practices (MacIntyre, 1981). It is from this link that we draw our framework: Individuals are motivated because through participatory exposure to social practices, they learn what it makes sense to do; and, vice versa, by reflecting on the unity of life they shape social practices. This interdependence lies at the core of MacIntyre's virtue ethics (MacIntyre, 1998b) as it allows for a characterization of the virtues in the interplay of individual life and social practice.

	Self-determination view	*Social practice view*
Output	Product	Good
Incentive	Reward	Unity of life, moral obligation
Interaction with peers and tasks	Situational, next step, solution-oriented	Developmental, sequential, quest-oriented
Quality perception	Use-value	Standards of excellence

Table 19: Assumptions about the individual OSS developer

Following this logic, OSS developers are motivated by a consistent stream of actions that we call unity of life. Here, it is not the immediate and isolated outcome that matters (the carrot), but how the individual subjectively holds outcomes and actions to be consistent over time (the journey toward the end of the rainbow). We contrast the alternative set of assumptions with the assumptions observed in the reviewed literature. The dominant role of self-determination theory informs the logic of what we call the self-determination view.

The reviewed literature on OSS development motivation has not accounted broadly for the insight that social practices strongly relate to people's motivation (Morgeson and Humphrey, 2006). As a response we suggest a social practice view of OSS development, where individuals take a long-term, frequently developmental rather than situational perspective on social interaction, and pay attention to the importance an activity assumes for unity of life. Living well, in the Aristotelian sense, aims at the right timing of actions, attention to particulars, and aesthetic values (MacIntyre, 1981; Nussbaum, 1985). The individual views actions, outcomes and interactions through a "temporal lens" in order to achieve a sequence of events that supports a life well lived. A good life refers to a moral position that results from the accumulation of individual perception and more general principles (Nussbaum, 1985, p. 524). The general principles (standards of excellence are of particular interest here) are shared by members of the social practice, and, following MacIntyre, the individual gradually adapts standards of excellence developed in the social practice, changing the institution to become supportive rather than constraining, and producing a "good" in terms of these standards, rather than a product measured in, for example, features or hours spent working. To be clear, a good can only result from work, effort, and activities carried out in a social practice. Goods are collective in production even when only one person carries out coding activities, because coding follows standards of excellence that are defined by the collective, i.e. the social practice. In a social practice view, the individual's ideas of quality extend beyond use-value—the value for users in resolving their own technical problem—to account for what the social practice values and defines as the current state of the art.

To become an OSS developer means to acquire an identity related to a social practice. This means, contrary to the self-determination view, that an individual does not become an OSS developer simply by submitting code to a software repository. Martin (2002, p. 556) suggests: "These (social) practices enter centrally into defining a way of life, with technological development entering even more centrally into the ways of life of engineers. (The technologies are parts of or aspects of ways of life.)" Identification instills a sense of moral obligation in individuals to support and further develop the social practice. For example, a "moral obligation" in OSS is the early release of software patches to the public, an integral part of the social practice of OSS development (Raymond, 1999).

Personal commitment goes beyond commitment to the software product or external goods like profit but extends to the social practice and its broader social effects. From this perspective, OSS development embedded in a social practice becomes meaningful and a way of life with its own pleasure, challenges, and other benefits. Several authors have suggested that self-reflection helps shape practitioners' character (Flyvbjerg, 2001; Habermas, 1988; MacIntyre, 1981). Once developers contribute to the social practice, are motivated by it, and reflect on their actions (Calhoun, 1983), they also gain insights about their values in relation to that social practice, for example, whether or not the virtues it values fit their own personal virtues.

An orientation toward a sequence of tasks mirrors MacIntyre's idea about the individual's search for a narrative structure in life, which means that the sum and choice of our activities form a consistent story of our life or the role we see ourselves fulfilling. This can mean that a developer follows a quest that extends beyond the next step in a workflow or the solution to a specific task and accepts high goal ambiguity. Individuals are important actors who participate in building institutions that structure and govern the social practice (Geiger, 2009; Whittington, 2006) and whose motivations range widely from seeking autonomy, freedom from conventions, creativity, to the wish to change their world (Ketchen, Ireland, and Snow, 2007; Rindova, Barry, and Ketchen, 2009).[71]

In the social practice view, motivation can be directed at the good—which includes, e.g., soft-

[71] Many authors on entrepreneurship therefore understand motivation and social practice as inseparable (Steyaert, 2007, p. 467; see also Schatzki, 2005).

ware code produced following standards of excellence defined by the social practice—or at institutional change aimed at supporting the social practice. In either case it is not directly (short-term) reward-oriented but induced through the social practice. Hence, motivation in the social practice view differs from motivation in the self-determination view. According to the social practice view, motivation is intimately linked with a developer's experience of being a member of the social practice of OSS. One important result of this is that if people sense that the social practices they value have become corrupted in some way, for example by institutional goals, they often seek to change institutions.

These two views of motivation are complementary in the study of OSS development. Starting with the premise that the social practice view accounts for at least some individuals' behavior, the framework's theoretical conjectures can be used to formulate concrete propositions about OSS development and contribute to an agenda for future work. We begin the next section by illustrating this with an account of the emergence of the Free Software Foundation.

On the Emergence of an Institution

In the 1960s and 1970s much software development was carried out by scientists and engineers in academic and corporate research laboratories. It was a normal part of developers' social practice to give and exchange software they had written, in order to modify and build upon each other's software, both individually and collectively, and in turn to make their modifications freely available. Virtues such as the openness to sharing knowledge actively and intensely were considered important for learning, efficient code development, better bug-free products, and, overall, the development of the software engineering profession.

The emphasis on the virtues of sharing work in software development was very strong among a group of developers at MIT's Artificial Intelligence Laboratory during this period (Levy, 1984). The first conflicts between the institutional goals and the virtues of individuals valued by the social practice can be traced back to the 1980s. At this time, MIT decided to license some of the code created by this group to a commercial firm. In accordance with its commercial interests (external goods), the firm restricted access to the source code for that software to exclude the original MIT developers, creating a great deal of frustration and irritation among them. This incident is a good illustration of the kind of conflicts to which MacIntyre draws our attention. Whereas software developers at the time considered virtues such as openness, learning, and knowledge-sharing a prerequisite for the social practice's creation of internal goods (including product excellence, professional development, and benefits for others), they felt institutional concerns for the constraint of external goods, which they feared would eventually harm the practice. Richard Stallman, at the time a programmer at MIT's Artificial Intelligence Laboratory, was distressed by the institutional pressure to restrict access to source code and sell software through licensing. He believed it would harm the software engineering profession and hinder "humanity's rapidly growing need for better and better technologies." Stallman viewed these practices as "morally wrong" impingements upon the rights of software users to learn and create freely. In his own words, faced with the collapse of his community's social practice, he was had to make "a stark moral choice"(Stallman, 1999, p. 19):

With my community gone, to continue as before was impossible. … The easy choice was to join the proprietary software world, signing non-disclosure agreements and promising not to help my fellow hackers. Most likely I would also be developing software that was released under a non-disclosure agreement, thus adding to the pressure on other people to betray their fellows too. I could have made money this way, and perhaps amused myself writing code. But I knew at the end of my career, I would look back on years of building walls to divide people, and I feel I had spent my life making the world a worse place ... So I looked for a way a programmer could do something for the good. I asked myself, was there a program or programs I could write so as to make the community possible again?

Stallman's response to this challenge was to create an institution as an alternative to the firm, called the Free Software Foundation. The purpose of the foundation was to preserve free access to software developed by people who shared the virtues valued by the social practice. The legal mechanism he developed to support this idea was the GNU General Public License, which can be affixed to a piece of software by a developer, and which guarantees a number of basic rights to all future developers and users. These include the right to download for free, study, and modify the source code, and the right to redistribute to others modified or unmodified versions of the software for free. Stallman firmly believed that this

license and the new institution of the Free Software Foundation would support the social practice of software development and eventually help create excellent products of benefit to society. His institutional alternative was created to preserve the social practice's capacity to create internal goods alongside external goods pursued by the software industry. Later observers refer to Stallman's institutional alternative as a "free software" ideology that motivates developers to join and contribute to OSS (Stewart and Gosain, 2006). However, the origin of this collective action was the social practice of software development that motivated Stallman to create an institution. To paraphrase an insight from MacIntyre's theory, anything that does not promote the public good, such as the appropriation of collectively developed software code, may not be properly regarded as social practice.

An internal good can be a practitioner's pursuit of excellence in software development—it is against this backdrop we should understand Stallman's moves. Thus, the new institution of free software sprang out of practitioners' moral considerations, and these motives are shaped by the social practice of software development. Rather than ascribing to the view that the new institution originates in an overarching ideology that motivates people to act, we conclude that the concern for product quality, work, and the wider implications of Free and Open Source software gives rise to new institutions. This conclusion differs from other accounts of OSS, like Moody's Rebel Code (2001), or social movements that confront the establishment of the software industry on ideological grounds (see the discussion of ideology and collective action in von Hippel and von Krogh, 2003; Stewart and Gosain, 2006). Finally, consistent with the theory we develop, if the institutions of free and open source software fail to support social practices that create internal and external goods and enable developers' pursuit of excellence or unity of life, institutional alternatives will emerge that do the job better.

A social practice view on motivation in OSS development and issues for IS research

In the previous two sections we discussed, first, an alternative set of assumptions about individual OSS developers and, second, how the emergence of new institutions are linked to values central to the social practice of OSS development. Based on this we now develop a framework that links MacIntyre's conceptual building blocks with motivation. In this motivation-practice framework we consider OSS development the social practice under investigation. Thus, the motivation to contribute to OSS development can be rephrased as the motivation to contribute to the social practice of OSS development. However, in the MacIntyrian perspective adopted here, it is not meaningful to think of a social practice absent from institutions and goods. Strictly, a social practice cannot exist without producing internal goods; and it cannot exist without supporting institutions that protect its standards of excellence and enable the creation of external goods. Such is the nature of the social practice. Rather, a separation of the social practice from institutions and goods can only be made for purely analytical reasons, a fact we shall profit from when theorizing from our conceptual building blocks.

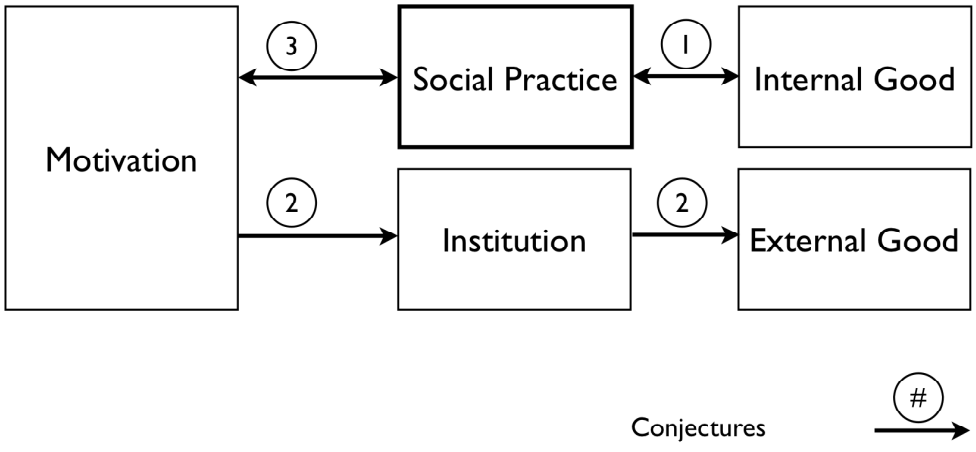


Figure 10: Relationships between motivation and social practice, institution, and goods.

We postulate that individual-level motivation to act is directly related to either active participation in the social practice or through attempts to change the supporting institutions of the social practice. Thus, goods (as outcomes and rewards) are not directly impacted by motivation. Rather,

the individual is moved by and through the social practice to contribute to the good or to change the institutions that support the social practice. Figure 1 depicts these relationships and shows three theoretical conjectures discussed in detail below. In its most basic form, the framework illustrates that researchers should not implicitly assume a single unidirectional link between motivation, action and reward. Thus, using the framework researchers can ask questions previously not considered to belong to the domain of motivation.

The integration of motivation with MacIntyre's conceptual building blocks creates relationships that suggest answers to the three questions stated at the outset of the paper, in the form of theoretical conjectures and concrete propositions. Each of the three questions corresponds to one of the relationships in the framework (numbered in Figure 1). Next, we discuss corresponding conjectures and propositions as well as issues for future research.

Theoretical Conjecture One

Question 1: How and why do OSS developers produce high-quality software (goods) when they do?

Theoretical conjecture 1: OSS developers contribute to the production of internal goods (e.g. high-quality software) when their actions follow the standards of excellence of the social practice. At the same time, the internal good produced by the social practice impacts on standards of excellence pursued in the social practice.

Proposition 1: Developers in a social practice create a sense of timing and developmental interactions with peers that improves the social practice.

Proposition 2: The software product impacts on the standards of excellence in OSS development.

First, quality is an essential element in understanding social practices because its members learn, internalize, and eventually improve the standards of excellence. Developers need to acquire an understanding and appreciation of quality by participating (for some time) in the social practice. In the framework, quality is not globally assessed or measured but is understood as internal to the social practice: it is defined as a characteristic of internal goods that adhere to the standards of excellence in the social practice. Revisiting the assumptions about the individual developer, Proposition 1 fits with the observation that participation in a social practice requires engagement with the practice's goals, a developmental understanding of social relations within the practice that leads to a broader set of activities (e.g., including helping behavior, see Lakhani and von Hippel, 2003), and an appreciation of incentives (their timing and moral and aesthetic value) beyond the immediate rewards of an activity. This is why OSS developers devote considerable effort to making code "beautiful," helping other developers to understand and appreciate the nature of "beautiful code," or releasing a new version of software at symbolic points in time. One OSS project, for example, chooses release version numbers that asymptotically approximate the number π.

In addition, Proposition 2 fits with the observation that OSS developers often receive immediate user feedback on installing, systems compatibility, bugs, license restrictions, etc. (Lee and Cole, 2003; Francalanci and Merlo, 2008). When developers see how the software product performs on their own and other users' computers, or compare in efficiency with competing products, they may choose to maintain or adjust the standards of excellence in the social practice, as part of learning to practice better. In OSS development, software users often communicate directly with software developers, exemplifying their intent with code patches. The issue tracker of the Firefox web browser, for example, is filled with feature requests that are often accompanied with source code patches implementing these.

The theoretical conjecture also entails a research agenda: What activities are considered good (e.g., code production, training of other developers, improving governance structures, or communication tools)? Collective work also involves mundane tasks that, nevertheless, are taken up voluntarily if need be (Shah, 2006). In our perspective, what needs to be done to achieve quality is determined collectively by individuals who "hold one another accountable to what is at issue and at stake in ongoing practices" (Rouse, 2007, p. 54). Yet, by which other mechanisms are quality standards collectively determined? Furthermore, what does the perception of quality, informed by standards of excellence, mean for firm-community collaboration? What are the sources of heterogeneity in standards of excellence across OSS communities? These two questions are related because individual contributors (such as the representative of a software firm) help define standards of excellence in their respective communities. In turn, the quality standards in an OSS community might have repercussions within firms that participate in collective development. The extent to which these repercussions lead to

conflict or to fruitful interaction between firms and communities is largely unexplored.

Theoretical Conjecture Two

Question 2: Why do OSS developers change institutions?

Theoretical conjecture 2: OSS developers change institutions when and where these institutions no longer protect sufficiently the standards of excellence of the social practice. In addition, institutions are changed in order to provide external goods that support the social practice.

Proposition 3: Institutions that offer external goods to OSS developers (such as firms participating in OSS and hiring developers) are judged as supportive or constraining by the developers according to the institutions' adherence to the standards of excellence defined in the social practice.

Proposition 4: Under certain conditions, developers of OSS are prepared to sacrifice potential rewards (external goods offered by a current institution) in order to make an institution compatible with the standards of excellence defined by the social practice of OSS.

Second, motivation may trigger institutional change. The earlier example of Richard Stallman, founding the Free Software Foundation, illustrated a form of motivation geared not at immediate rewards but at the protection of a social practice that achieves internal good. This can be labeled "ethical" or "ideological" and makes perfect sense when viewed from the perspective of the social practice. Institutional change was necessary to protect the social practice from extinction, given that newly formed software companies hired away Stallman's community of peers and threatened to change radically the way software was developed. The changes Stallman and his peers implemented in the way software could be licensed represented an institutional change. The license regime regulates the exchange of software among developers. The exchange is part of the social practice and free exchange is considered good practice in OSS development. MacInytre focuses on virtues that extend beyond the short-term interests of individuals giving social practices an enduring character worthy of analysis in their own rights (see also Halliday and Johnsson, 2010). In brief, we propose that these enduring characteristics of the social practice give rise to institutional change via the motivation of developers.

When institutions, such as firms, cannot sustain social practices sought by virtuous software developers, the latter will seek to change or develop new institutions that serve better the internal goods of the social practice. Proposition 3 builds on the argument that practitioners will judge institutions by their adherence to the standards of excellence of the social practice because a firm, for example, that contributes to OSS development engages in the social practice and offers external goods (such as salaries and career opportunities). As an output of institutions, external goods need to enable standards of excellence of the social practice that the institution supports. This could mean that salaries or promotions within the firm need to be perceived as fair according to the standards of excellence in OSS development.

In the motivation-practice framework, institutions serve the social practice and not the other way around. The priority of the social practice over institutions implies that individuals will sacrifice potential rewards offered by an institution (such as a firm hiring OSS developers and offering a salary) in favor of adhering to standards of excellence, as Proposition 4 states. Thus, the "good and right thing to do," according to the social practice's standards of excellence, may be to change the institution and risk losing external goods in favor of internal goods if standards of excellence are threatened, as the example of Stallman illustrates. This could mean for a developer to lobby for policy changes, implement corporate restructuring, or leave a well-paid job in a software firm.

Scholars currently cannot explain when institutional change is called for in more general and systematic terms, which opens a number of related research questions. For example: How are conflicts between extrinsic motivation and moral obligations resolved when current institutions impose a choice between the two (as in Stallman's case)? Under what conditions do the assumptions about motivation from the social practice view hold and account for institutional change? How can organizations accommodate individuals prepared to engage in institutional change in order to pursue standards of excellence? And related to this, how can organizations drive institutional change with the motivation of developers?

Theoretical Conjecture Three

Question 3: Why do developers sustain the social practice of OSS development?

Theoretical conjecture 3: OSS developers sustain the social practice of OSS development because social practice instills the motivation to uphold its standards of excellence over time.

Proposition 5: The motivation to contribute becomes stronger during developers' tenure in an OSS community and by their contribution to the social practice.

Proposition 6: Through sustained contributions to the social practice, developers become motivated to contribute beyond code patches to educate and help others, and take on tasks that support the internal good of the social practice.

Third, how does a social practice influence motivation and vice versa? Perhaps surprisingly, this question does not appear to be prominent in MacIntyre's work. Virtues are transmitted to individuals through socialization and collective work and exercised within social practices by individuals who choose activities that appear right in their life. They choose to sustain practices that cultivate internal goods that match their individual sense of the common good and the life they wish to live. Entry and exit represent the dynamics of these individual choices. Organizations continuously face the entry and exit of individuals who choose freely how and where to enact their profession. It follows from our argument that motivations change with the context of the social practice. A first entry into a community by making a contribution to OSS is motivated differently from a sustained contribution to the same practice: Shah (2006) observes that mundane tasks tend to be accomplished by long-term members of communities. This can be interpreted, in line with our Propositions 5 and 6, as a growing understanding of the internal goods of the social practice and a stronger motivation to sustain contributions. Developers gradually develop an understanding and appreciation of the larger needs of the practice and are motivated to contribute to necessary tasks that might appear mundane to the outside observer. Tasks such as bug fixing, maintenance of software, or helping new users are, however, essential for the social practice to function and produce high-quality software, perhaps understood even better by seasoned developers than newcomers. A flourishing OSS community succeeds in creating this understanding with its members and in motivating them to see these activities as core to the production of internal goods.

Future research should uncover more detail about mutual influence. The motivation to contribute to a social practice may change in quality as well as in direction and open questions relate to both possibilities. For example: Could a loss of interest in the internal good and subsequent exit be triggered by conflicts over incommensurable moral premises? The emergence of a proprietary software industry, as witnessed by Stallman, made him the architect of a new software license that regulates the free exchange of software. When do competing views of what constitutes a good practice jeopardize developers' long-term motivation? Under what conditions does an initial interest grow or wane over time, given exposure to the social practice, learning, social interaction, moral premises, help received, or a sense of reciprocity in contributions to OSS?

One individual's mark on the social practice may turn its course of action or significantly extend its standards of excellence. According to the definition we discussed above, a social practice is coherent and complex, two characteristics that might be explained by the content of the work and the history of the internal discussions of developers about the good it achieves. However, the social practice depends on individuals who carry it forward, understand its complex history and possibly incoherent tendencies, and correct and sustain the social practice going forward. This is easier said than done. So far, research on OSS has not contributed much to the understanding of individual motivation beyond the contribution of code itself, which raises a number of future research questions. For example: Why do individuals found new or fork (split up) existing OSS communities? Why do we observe a wide heterogeneity of communities and philosophies within OSS (Himanen, 2001; Moody, 2001)? There are, for example, distinct differences in ideology and values between pragmatic OSS enthusiasts and the Free Software Foundation (Stewart and Gosain, 2006). The wealth of licenses and approaches to similar problems calls for more research on why individuals choose different paths and subtle changes over the established wisdom found in some social practice (developers are motivated to build institutions, not only code). Given the chance, OSS developers may diverge, follow their own ideas, and realize similar if not identical solutions in different ways. Thus, what are the proper time intervals to study motivation if one is to consider motivation in which developers seek unity of life? Another interesting topic worth exploring is the differences between paid and unpaid developers. In the motivation-practice framework, paid developers partly receive monetary compensation provided by institutions, whereas unpaid developers need to seek other means of compensation. Future research could, for example, explore relationships between monetary compensation and behavior within the

social practice in order to identify potential threats to the sustenance of the social practice, beyond the contributions of individual developers. Finally, our review of motivation to contribute to OSS reveals a bias toward assumptions originating in self-determination theory, and prompts the development of a complimentary set of assumptions that we associate with a social practice view, building on the theory of MacIntyre. The assumptions underlying the social practice view raise the empirical question of the actual distribution of motivation types in a population. Future research may study the distribution of the two "ideal types" of individual in a distribution of OSS developers.[72]

[72] Methodologically, this could be done analogously to the study of social preferences in behavioral economics—for example, as Fehr and Schmidt (1999) have done in their experimental studies to determine the distribution of inequity-averse individuals in a population of students.

CONCLUSION

The objective of this paper is to provide a state-of-the-art review of the study of motivation to contribute to OSS development, and reinvigorate the research field by providing a new theoretical framework with propositions. We identify a large body of work examining types of motivation that lead developers to contribute their time and effort to the development of OSS. However, we argue that both the antecedents and consequences of motivation are more extensive and complex than the present level of theorizing and empirical research has suggested.

OSS development differs from conventional software development along three dimensions, incentives, control and coordination mechanisms, which in turn are reflective of a distinct social practice in which ethics plays a central role. Because of these differences, we argued that individual motivation should not be looked at in isolation. Instead, scholars should expand theory building and research to cover the interplay with institutions, goods, and the social practice. Following a trend to explore the role of social practices in the IS field (Kellogg et al., 2006; Orlikowski, 2000), and the social sciences more generally (Schatzki, 2005), we develop a theoretical framework around the conceptual building blocks of social practice, institutions, and goods. This new motivation-practice framework is based on the theory of Alisdair MacIntyre (MacIntyre, 1998b, 1981) and includes ethical considerations of social practice. The framework's set of theoretical conjectures and accompanying propositions provide answers to the three questions identified in the beginning of the paper and point to future research opportunities.

While the main contribution of the paper is the new motivation-practice framework, we also (a) formulate implications for IS management, (b) discuss the relationship between institutional change and social practice, and (c) tie implications from the framework back to the self-determination perspective in motivation studies.

Several implications for IS management can be extracted. First, standards of excellence emerging in global communities of software developers can gain broad endorsement and impact quality standards expected by users and customers. For example, peer review and quick feedback loops in OSS are said to lead to better quality. Installing a culture of "doing things properly," rather than quickly hacking around bugs may ultimately result in better code quality. Finally, established tools developed for and within OSS can be employed by any software development project. Hence, close observation of the standards of excellence that characterize OSS development may pay off also for software firms that do not actively participate in OSS development.

Second, software developers collectively account for and evolve the virtues in social practice. MacIntyre (1981) refers to the social practice as a "school of virtue," because individual learning is shaped by and informs the collective appropriate conduct regarding ways of developing software. Participating in OSS development could thus provide a valuable training ground for software engineers, in so far as the firm sees fit and is prepared to comply with the ethical considerations in the social practice. In this regard, the new framework informs managers about the relationship between OSS and proprietary software development. OSS does not necessarily exist in competition to proprietary software, but rather complements it; it secures the social practices through which certain standards of excellence in software development can be further nurtured. Software firms are actively looking for developers who have "cut their teeth" in OSS communities; firms such as Intel or Red Hat recruit developers whose skills and standards of excellence stand out.

Third, as shown in the framework, incommensurable moral premises may collide within a firm when developers adhere to different traditions yet collaborate on the same projects. In a social practice perspective, one may identify and preemptively solve looming conflicts over technology, standards, rules, and routines in software firms. However, the activity of software development often builds on and refers to earlier work and developers and the firms they represent may be held accountable by other OSS developers for the work they submit to OSS projects. Questions of compatibility and compliance with OSS development play an increasingly important role for those firms that develop both proprietary and open source software.

Fourth, a related issue is that the use of incentives and control in software firms may need careful tailoring to fit with developers' motivations to contribute to OSS development. Monetary incentives are of course compatible with OSS de-

velopment (Roberts et al., 2006), but social practices can instill loyalty and lead to motivation for institutional changes that perhaps question the efficiency of such incentives (Figure 1).

Fifth, developers can be motivated to change institutions in order to support their practices. This may apply to firms, too. For example, IBM founded the non-profit Eclipse foundation as a neutral steward for technology that IBM had initially developed, and created formal membership roles in which IBM was only one among equals, in order to facilitate external contributions of other firms and individuals. Intel and Oracle have founded institutions such as the Open Innovation Network, which grants mutual licenses and a non-litigation agreement for certain OSS technologies to all of its members.

The theoretical framework developed here also opens up a new view on the relationship between institutional change and social practices. Other perspectives on the emergence of institutions (see Hargrave and Van de Ven, 2006, for a review) argue convincingly that the struggle between factions and social movements brings about change and institutional innovation. However, framing and political struggles may capture neither the full diversity of people's motivations to contribute nor the role technology might play in this process. As the case of the Free Software Foundation demonstrates, notions of quality that drive social practice to generate and maintain its standards of excellence can be powerful forces for change. These views are compatible, in that a social practice of OSS development can become a social movement (Hertel et al., 2003; von Hippel and von Krogh, 2003). They are also complementary, in that collective action emphasizes the "struggle over meanings of new issues and technologies" (Hargrave and Van de Ven, 2006, p. 884) across social movements, whereas social practice focuses on the quest for higher quality products, more profound knowledge, and improved collaboration in creating internal goods.

Collectively, we have merely begun to scratch the surface of a full exploration of developers' motivations for contributing to OSS development. There are vast research opportunities in all areas covered in this theory and review piece. The motivations across OSS projects could, for example, be investigated productively using self-determination theory. Three topics in particular could stimulate interesting research from a self-determination perspective. First, future research can incorporate ethical considerations in self-determination models, since ethics is integral to the OSS phenomenon per se. Self-determination theory would suggest the existence of both intrinsic and extrinsic ethical motives. Disentangling these ethical motives and linking them to types and levels of involvement would be an important contribution. The motivation-practice framework, in turn, suggests that ethical considerations are geared toward the virtues and standards of excellences of the social practice. For example, self-determination research could fruitfully explore how developers are intrinsically motivated toward maintaining standards of excellence. Again, the motivation-practice framework is complementary since it explains how standards of excellence are formed collectively through active participation in the social practice over time.

Second, our argument that social practice moves individuals to act may inform future applications of self-determination theory in two ways. Ethical considerations may not only directly motivate individuals to act, but also condition what is self-determined. Cultural analysis in social sciences recently shifted to treating culture "as constitutive of a wide range of social processes rather than a regulative that works against other forces, such as interests or rationality" (Weber and Dacin, 2011: 287; see also Boltanski and Thévenot, 2006). Motivation studies using self-determination theory should explore if and to what extent a contextual and cultural substrate impacts the confines of what individuals experience as self-determined action. Further, the social practice perspective suggests a carefully paced integration of temporal aspects to context when designing studies using self-determination theory. In other words, the perspective points to potential changes in motivation over time triggered by exposure to and integration in a specific context. Self-determination models in the reviewed literature treated context as exogenous and static and thus not accountable for motivational dynamics. For example, self-determination research could investigate motivational differences between groups of developers with different lengths and types of contextual exposure to OSS projects.

Third, the understanding of how OSS institutions (e.g., standards, licenses, governance, copyright and IP assignments) and supporting umbrella organizations relate to developers' self-determined motivations is only beginning to emerge and many questions remain unanswered. With the growth of social networking sites and online communities (Wiertz and de Ruyter, 2007;

Wasko and Faraj, 2005) countless other practitioners apart from software developers rely on volunteer contributions and institutions such as open content licenses. Obviously, motivation research on the OSS phenomenon may inform research designs in other domains, and vice versa.

To conclude, while there is ample room to investigate motivation in OSS from a multitude of perspectives and methodological approaches, we believe the greatest research opportunities lie in those questions found at the intersection between social practices and institutions, against which individual motivations can and should be understood. Figuring out what moves people, we should start with the assertion that people's pursuit of visible carrots is at times interrupted by the larger quest for the invisible gold at the end of the rainbow.

MODDING AS RATING BEHAVIOR IN VIRTUAL COMMUNITIES:

THE CASE OF ROOSTER TEETH PRODUCTIONS

Stefan Haefliger
Philip Reichen
Peter M. Jäger
Georg von Krogh

Chair of Strategic Management and Innovation
Department of Management, Technology, and Economics (MTEC)
ETH Zürich, Switzerland

Published in Lecture Notes in Computer Science vol. 5621 2009 pages 197-206

Virtual communities that make use of social network site features blend known applications of virtual communities. These communities can be simultaneously social and commercial, organization sponsored and heavily relying on member interaction. We explore modding behavior that allows members to evaluate other members' contributions both with numerical value and qualitative rating. We show that approximately half of all members received mods on their comments, that the majority of mods given were positive, and that the amount of mods received for a comment was related to the position of the comment in the community website's thread. Contributing to the emerging literature of social network sites and virtual communities, we discuss implications for theory, future research and management.

INTRODUCTION

In a recent MIT Sloan Management Review article, Bernoff and Li (2008) suggested "People are connecting with one another in increasing numbers, thanks to blogs, social networking sites like MySpace and countless communities across the Web. Some companies are learning to turn this growing groundswell to their advantage." With close to one billion[73] people connected to the Internet, firms not only face unprecedented opportunities but also considerable threats in such a digital economy. Numerous firms have set up "virtual communities," a term coined by Rheingold (1993). These communities are, mostly but not exclusively, online spaces in which customers and non-customers can interact with the firm and each other. Porter (2004) defines virtual communities as an "aggregation of individuals or business partners who interact around a shared interest, where the interaction is at least partially supported and/or mediated by technology and guided by some protocols or norms." (see also Porter and Donthu, 2008).

Virtual communities can have a positive impact on firm performance. According to one study, revenues have increased more than 50% for some firms (Algesheimer and Dholakia, 2006) that have managed these communities well. In addition members of virtual communities remain twice as loyal to and buy almost twice as often from the sponsoring firm. Armstrong and Hagel (1996) found that "companies that create strong online communities will command customer loyalty to a degree hitherto undreamed of and, consequently, will generate strong economic returns". In addition, virtual communities can shift bargaining power from suppliers to customers (Kozinets, 1999); spread positive word-of-mouth (Dholakia, Bagozzi and Pearo, 2004); help firms learn from customers (Kardaras, et al., 2003); increase website traffic (Hagel and Armstrong, 1997); raise entry barriers for competitors (Hagel and Armstrong, 1997); facilitate product development efforts (Nambisan, 2002); and increase customer satisfaction and loyalty (Shankar, et al., 2003).

Recently, social network sites caught the attention of users, firms, and researchers (Boyd and Ellison, 2007). Sometimes labeled "Web 2.0" coined by Tim O'Reilly[74], social network sites (SNS) emphasize member profiles and direct interaction and links between members, provide content ratings, and enable rating behavior (de Valck et al., 2007; Kim et al., 2008; Pascu et al., 2007). "Modding" (derived from "moderation") refers to a type of trust rating that "allows members […] to evaluate other users' reviews with numerical ratings" (Kim et al., 2008: 532). Modding is a direct feedback mechanism between community members.

Both streams of research on virtual communities and social network sites belong to the field of computer-mediated communication. The combination of features within one online environment triggered new forms of behavior that warrant analysis. If virtual communities make use of SNS features the combination results in a new type of virtual community that cannot easily be understood by the frameworks used to classify virtual communities (Porter, 2004).

A virtual community that makes use of social software features may be organization sponsored, yet dominated by direct interaction among community members, hence, social and at the same time commercial. SNS features offer communication structures that make member-to-member communication easier and more frequent. Moderated communication makes members become more socially embedded in the virtual community (de Valck et al., 2007; Algesheimer et al., 2005). The availability of new communication structures that allow direct feedback on contributions calls for research exploring rating behavior. Specifically, modding of member comments by other members extends the communication options usually associated with virtual communities and call for more research on mass communication in virtual communities (Schoberth et al., 2006). Schoberth and colleagues (2006) found, among other things, heterogeneity in community participants' activities. Scholars have also called for more quantitative research using behavioral data from virtual communities (de Valck et al., 2007; Casalo et al., 2008). Thus, we ask: how do members of a virtual community make use of modding?

73 Source: http://www.comscore.com/press/release.asp?press=2698

74 For more details: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

METHODOLOGY

We conducted a large quantitative study on the virtual community of Rooster Teeth Productions, a Machinima Production company creating and publishing animated videos made in computer games. We present the sample case as well as data gathering and analysis in this section.

2.1 Sample Case

Producing animated videos was previously restricted to media professionals because of the high cost of software packages. These restrictions led innovative users to produce animated shorts with computer games by using the underlying 3D render technology thus creating Machinima. Game engines were relatively cheap compared to traditional production tools. In addition, most of the in-game assets like characters and landscapes were already at hand, which reduced the overall production time for an animated movie significantly.

Rooster Teeth Productions is one of the most successful Machinima companies (von Krogh et al., 2009). They sell sponsorship subscriptions, merchandising, and DVDs and reach a large user community. The latter, in fact, was triggered early on when Rooster Teeth introduced an elaborate community platform offering SNS features:

> *"… well, I think a lot of it has to do with the fact that the community site that we have made … or at least at the time we made it … had features that weren't that present in other places, we were a little ahead of the curve at that time, and so there were a lot of cool features that people were interested in. This is like before MySpace really had taken off … So we've always tried to give it a little functionality, things they do in a community website they're interested in making … you know, interested in being a part of it. We tried to make the website almost like a game." Geoff Ramsey, Rooster Teeth Productions*

Rooster Teeth Productions was founded in 2003 by Burnie Burns, Matt Hullum, Geoff Ramsey, Jason Saldaña, and Gus Sorola in Austin, Texas. Their first and most widely known Machinima production was Red vs. Blue (RvB), a show featuring two teams of soldiers in the game Halo who are stationed in an isolated canyon where their sole purpose is to fight each other. The popularity of the show that first aired April 1st, 2003 profited from the humorous dialogues between the different characters. While the comedy was first aimed at other gamers, a broad audience swiftly appreciated RvB. To date, Rooster Teeth has released five seasons of RvB 'The Blood Gulch Chronicles,' and one season of RvB 'Reconstruction' comprising 20 to 25 episodes each as well as several spin-off mini-series. Over the years, shooting the movies has advanced from the game Halo 1 on the xBox to the latest release Halo 3 running on xBox 360 with overwhelming new possibilities in graphics and artistic composition. In addition, most of their merchandising articles were related to RvB, which remained the flagship show. Apart from RvB, Rooster Teeth produced several other shows including 'The Strangerhood', 'P.A.N.I.C.S.', or '1-800-Magic', using different game engines to shoot the films.

Each series had its own website on which the videos were shown, important announcements from Rooster Teeth staff members published, and where fans discussed topics around the show. The discussion took place where the videos were viewed – especially while viewers waited for the download to finish or directly after watching videos online. Users did not have to actively go to a website to express their thoughts about the product as is the case with most websites of communities of consumption.

Due to a steadily growing fan base, over the last four years the segment of the RvB community actively contributing to discussions grew to 42,000 members who posted more than 400,000 comments on 165 episodes. Members could choose their level of engagement. They could be mere "consumers" who just watched the videos and/or bought merchandising products without interacting, or they could interact with other community members. The Rooster Teeth community cannot be neatly classified as either VC or SNS since different users engaged differently.

With 16% of all members, the 18 year olds represented the largest group (see figure 10). The aver-

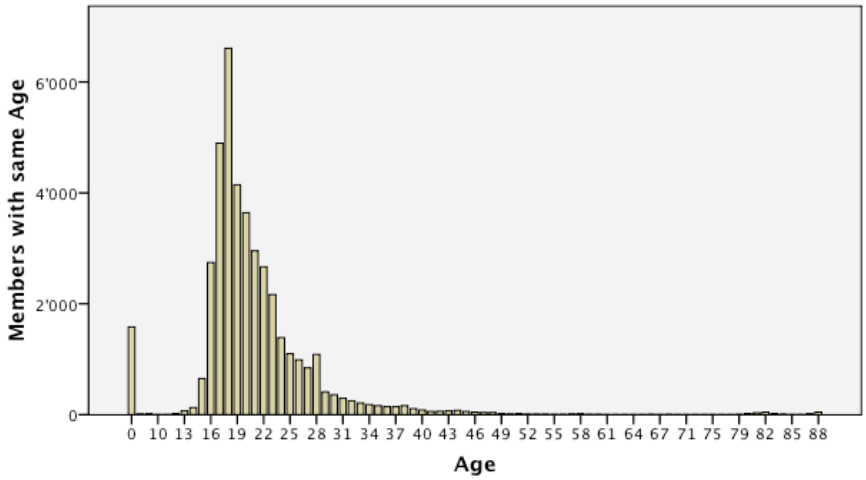


Figure 11: Age distribution of members commenting on Red vs. Blue

age age of members was 21 years with a standard deviation of eight years. The age distribution was biased and positively skewed by the fact that members who didn't enter any age were listed as zero, and that few members who apparently entered the maximum age of 88 years. 93% of the members were under 30 and the bulk was either in high school or college-age.

Tracing the amount of members over four years, we found that the community had been growing at different speeds, but steadily in volume in a nearly linear fashion (r square= 0.95). There were four visible gaps in signups, which were located in the first two years of its existence with the longest gap lasting for two weeks (see figure 2).

The amount of comments per episode varied from a minimum of 58 to a maximum of more than 28,000 with an average of 2,400 comments per video (Figure 12). Five different sections could be identified with a strong cyclicity given that the amount of comments increased notably during seasons: Section 1 represents the comments to Season 1 and 2 that were aired on a former version of the Rooster Teeth community website. Those comments were not migrated to the new and more elaborate software infrastructure and therefore the amount of comments was low in section 1. Section 2 followed the launch of the new website before Season 3 leading to a steep increase in comments eventually coming to a slowdown after the end of the season. Section 3 was the most commented section ever covering Season 4. Half of the total comments-population was found in this section. This finding does not imply that all comments were made during Season 4 since it was possible to comment 4 and 5 covering Season 5 and the start of RvB 'Reconstruction' respectively contain again relatively little commented products.

The basic units of analysis were the RvB-related comments made by members and the mod-points associated with the comments. These were displayed chronologically below the corresponding video similar to YouTube with the difference that the comments in our case were ordered by ascending post date (i.e. the oldest post was displayed first). All comments and the associated mod-points were publicly accessible. In order to leave a comment one had to be signed in as community member. Member accounts were free of charge and did not have to be activated by a moderator or an administrator. Hence, members were able to sign up at any time and start posting.

Comment modding is the act of rating another member's comment(s). Synonyms are 'rating', 'giving mod-points' or simply 'modding'. In the Rooster Teeth community each modding of a

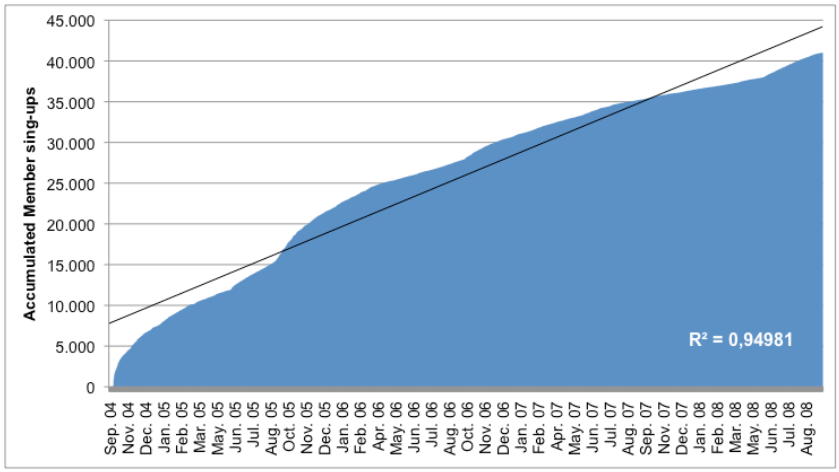


Figure 12: Accumulated daily sign-ups of members over the last four years.

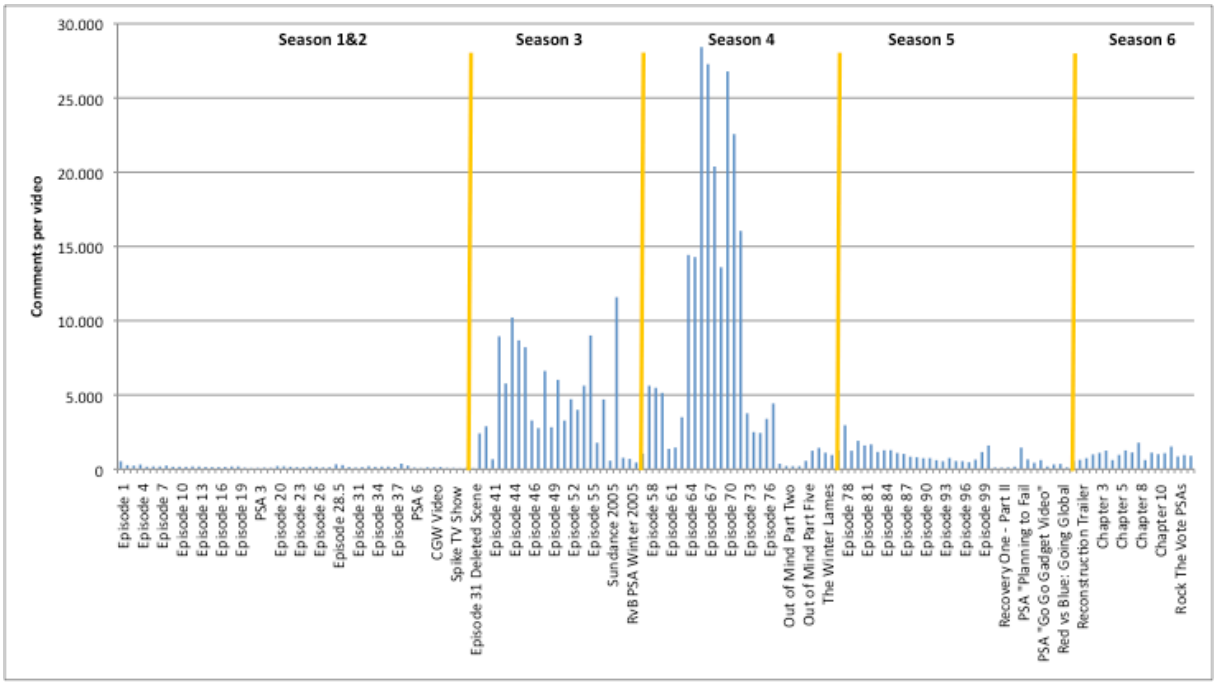


Figure 13: Comments per video. Episodes ordered chronologically.

comment consists of a combination of two values: a numerical value and a qualitative rating. The numerical value is either '+1' or '-1'. Each numerical value has to be combined with one of four qualitative ratings from which users can choose in a drop down menu. The four qualitative ratings corresponding to '+1' are 'Cool, Ditto, Funny and Zing![75]'. The four qualitative ratings corresponding to '-1' are 'WTF[76], Lame, Flamebait[77] and Noob[78]'. Mod points could only be given once per user and per comment. A user who has one account can mod each comment by another user only once. The mod is then publicly displayed next to the comment.

2.2 Data Collection and Analysis

For the purpose of the quantitative data analysis, we built up a database dedicated to the case under study. All available data from the Rooster Teeth RvB website concerning the episodes, the members, and the comments was automatically fetched during a three-day period from September 20th to September 22nd, 2008 and transferred to a local MySQL database for further analysis. To be granted full access to all the data, we obtained a sponsorship account. After screening and evaluating the data, we discovered some missing data sets that had been left out due to server maintenance by Rooster Teeth. For this, we obtained the missing data sets on October 2nd. All data entries in the local database indicate their fetch time stamp to check for possible inconsistency. We rebuilt the relational database structure of the original website using a separate table for episodes, members, and comments which were linked by their dataset identification number 'id' that remained the same as the online PHP web queries.

We fetched a total of 42,771 member accounts and 483,272 comments with their corresponding information. Out of all 737,000 registered Rooster Teeth community members[79], only those

75 Three possible definitions for our purpose: 1) New term for "owned", said after saying something witty to someone in an insulting manner. 2) If someone makes an absolutely awful joke, or says something completely random or pointless. One member of the group may "zing" them. 3) A noise made when a person, place, or thing is discriminated against in a humorous manner. Source: www.urbandictionary.com

76 Internet slang acronym for "What the fuck?"

77 A message posted to a public Internet discussion group, such as a forum, newsgroup or mailing list, with the intent of provoking an angry response (a "flame") or argument over a topic the troll often has no real interest in. Source: www.wikipedia.org

78 Short for "Newbie". A slang term for a newcomer to online gaming or an Internet activity. Source: www.wikipedia.org

79 http://rvb.roosterteeth.com/members/stats/

who at least commented once on a video of RvB were considered. Cleaning the fetched data sets from invalid information (either comments which link to a NULL member id or comments which link to empty member profiles) left us with 406,173 comments and 41,016 user profiles (see Table 17).

SPSS, Excel and the phpMyAdmin interface of the local server were used for the quantitative data analysis.

Almost half of all members posted at least one comment, which has been modded, but only 15% of all comments were modded (see Table 17). One possible explanation could be information overload (de Valck et al., 2007). Members cannot browse the overwhelming amounts of comments that are posted. Observing modding behavior in more detail, we find that 60% of the modded comments obtained positive values (cumulated mod rating > 0), 36% obtained negative values (cumulated mod rating < 0), and for 4% of the comments the mods evened out (cumulated mod rating = 0). The fact that 60% of all comments carried a positive rating, with "Cool" being the predominant rating class, showed that members generally tended to give friendly mods.

	Community Members		Posted Comments		Description
Total in Data Sample	**41.016**	**100,0%**	**406.173**	**100,0%**	**User-based and RT-members**
un-modded	21.426	52,2%	346.594	85,3%	Comments that were never modded
modded	19.590	47,8%	59.579	14,7%	Comments that were modded at least once
Total modded	**19.590**	**100,0%**	**59.579**	**100,0%**	
In sum positively rated	14.330	73,1%	35.363	59,4%	All comments with rating >0
In sum negatively rated	9.491	48,4%	21.638	36,3%	All comments with rating <0
In sum 0-rated	1.871	9,6%	2.578	4,3%	All comments with rating 0
Zing! (+1)	2.110	10,8%	3.076	5,2%	All comments with rating "Zing!"
Cool (+1)	6.308	32,2%	13.053	21,9%	All comments with rating "Cool"
Ditto (+1)	8.080	41,2%	12.832	21,5%	All comments with rating "Ditto"
Funny (+1)	4.743	24,2%	8.365	14,0%	All comments with rating "Funny"
WTF (-1)	3.301	16,9%	4.952	8,3%	All comments with rating "WTF"
Lame (-1)	3.850	19,7%	5.956	10,0%	All comments with rating "Lame"
Flamebait (-1)	2.803	14,3%	4.812	8,1%	All comments with rating "Flamebait"
Noob (-1)	4.075	20,8%	6.532	11,0%	All comments with rating "Noob"

Table 20: General statistics for modding behavior in the Rooster Teeth online community.

Next, casual observation suggested that modding behavior centered around early comments. We analyzed the attention different comments received based on their position in the comments thread (see Figure 13). This position corresponded to the time the comment had been posted in ascending order i.e. the comment that was posted first is at position 1, second at position 2, and so forth. Post numbers are displayed on the x-coordinate. For the y-coordinate we defined and calculated a ranking variable. We summed [0 to 36,921] the absolute values [0 to 2,920] of all mod points given to comments which share the same position (post number) [1 to 32,780][1] and divided this value by the amount of modded comments [1 to 165][80] per position. We thus calculated the average mod points per comment position. In effect, we used the absolute mod value for a better representation of the attention a comment received, than the net value.

We then ordered this quotient by descending value, i.e. starting with the highest value in order to receive a non-scaled ranking. For example, the data point at y=1 is calculated as follows: We considered all comments with post number 1. Since all 165 videos were commented at least once, 165 comments resulted. Some of these comments appeared to have invalid data base entries on the website. After pruning those, we were left with 147 valid comments. Out of these 147 comments we only examined those that were modded. In post number 1, all 147 were modded. We then summarized 147 absolute mod values, and divided the sum by 147, resulting in 36,921/147= 251. Repeating this procedure for all comments that were posted second (position 2) we get 107 respectively. Next, we ordered the quotients by descending value and displayed them as ranking. Thus, the y-axis represents the 'attention' or the valuation (negative or positive) members accorded a comment where 1 is the top rank with the highest attention.

[80] The maximum amount of 165 videos that could be commented on limited this number.

The results show that the first comments on each video received on average more (absolute) mod points than the respective subsequent comments. This holds true to a certain comment position from which on the data points become scattered. A threshold seems to appear around post number 50. The relationship between the post number of a comment, that is its position, and the mod value it received on average was positive and statistically significant (beta(14665) = 0.578 ; $p \leq 0.01$).

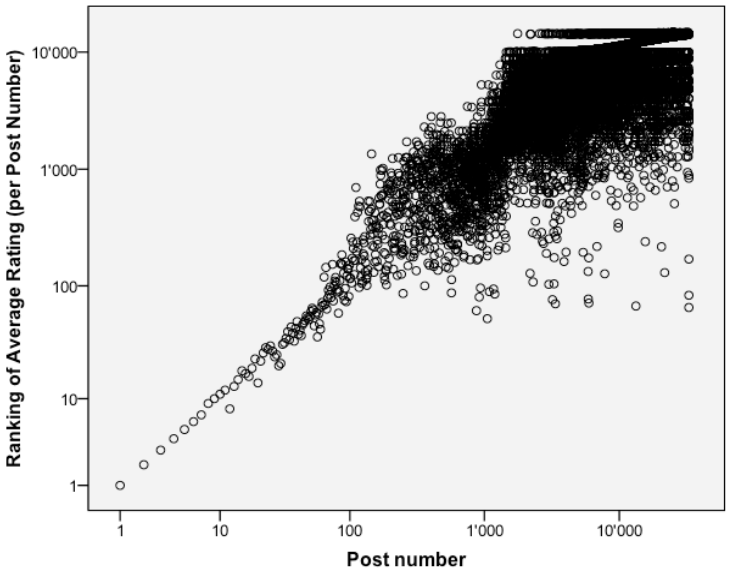


Figure 14: Relationship between the order in which comments were posted (ascending) and the rank based on the average (absolute) mod value the respective comment received (ascending: average to rank inversely proportional): both scales were logged.

Discussion

This exploratory study of modding behavior in a virtual community revealed three findings that open up for future research. First, just under half of the community members received mods on their contributions, while the other half did not received mods. Looking at the entire volume of comments, only 15% were modded. Second, the community studied leaned towards 'positive modding', with 60% of all mods being positive. Third, the time and location of a comment mattered strongly for the likelihood that it would be modded. Comments that appeared early after the release of a new product and appeared on the first two pages of comments, received disproportionately high amounts of mod-points. After approximately 50 contributions, the direct link between the position and the rank of mod-points weakened.

These findings warrant further research on virtual communities with SNS features in three areas: individual behavior, collective behavior, and community structures. First, roughly half of the community members never receive mods on their comments. The behavior does not seem to catch on throughout the member base. The extension to mass communication in virtual communities provided by modding seems to be used unevenly. Hence our results extend the findings of Schoberth and colleagues (2006) on heterogeneous communication behavior in online communities. Future research should analyze the factors that explain this behavior. Is modding considered to be costly, either in giving or in receiving? Is modding contested? Do member demographics explain modding behavior? Further, how does modding impact on contributions? Do members who received negative mods learn or change their behavior? Do positive mods (or mods at all) induce participation?

Second, we observed a friendly community who distributed more positive than negative mods. This result may impact on community growth, the willingness of members to contribute, and ultimately, consumer behavior. The result could support the idea that mods express trust rather than distrust or disapproval (Kim et al., 2008). What explains this bias? How does this finding compare to other communities' behavior? Can communities that lean towards offensive behavior be sustained? Also, researchers should conduct longitudinal studies of posting and modding behavior in order to identify changes in community behavior over time.

Third, comments that appeared early and high up in the (chronological) list of comments received disproportionately more mods than subsequent, less visible comments. A first interpretation suggests that community members may suffer from information overload and pay far less attention to later comments than to early comments. This finding raise doubts regarding the high expectations by some authors attached to modding and rating systems as quality signaling or filtering tools (Deng et al., 2008; Yang et al., 2009). Should the timing and position of a comment matter more than its quality in predicting the number of mods received, the modding system may be of little use to managers, marketing experts, and users of virtual communities. However, this issue needs much more investigation in future studies. We observed that after a certain threshold the post number did not predict the number of aggregate mods received. This calls for a refined analysis across multiple contexts and communication structures. Is it important that the first page contains 30 comments? Does the chronological order matter or could it be reversed and produce the same pattern?

Managers of virtual communities and social network sites may take away three insights from our study. First, virtual communities gain significantly new characteristics by adopting features associated with social network sites. Managers may think of more effective ways of distinguishing communities, possibly based on posting or modding behavior by members. The case of Rooster Teeth Production provides evidence as to the successful combination of product feedback and social network site features. Community members comment on a firm's products when they are released. They evaluate each other's comments and make use of the social infrastructure provided. Second, the modding behavior confirmed the impression of a friendly community. While this is only a first, preliminary finding it shows that the option of modding other community members' contributions was being used in a 'productive and supportive manner'. In general social network site features could be meaningful extensions to existing virtual communities. Third, fil-

tering valuable comments with the use of member-based modding tools may not be a simple matter. Our results show that only after about 50 comments the mods received started to deviate from the comment number as received chronologically. This may mean that after the first rush by people to make their comments visible, perhaps later mods may signal high-quality comments.

PARTICIPATION IN INTRA-FIRM COMMUNITIES OF PRACTICE:

A CASE STUDY FROM THE AUTOMOTIVE INDUSTRY

Patricia Wolf
Sebastian Spaeth
Stefan Haefliger

Department of Management, Technology, and Economics (MTEC)
ETH Zürich, Switzerland

Published in the Journal of Knowledge Management. 2011, issue 15, pages 22-39.

Purpose – Communities of practice (CoPs) have been found to support knowledge creation by enabling knowledge sharing among experts in firms. However, some perform better than others. This paper seeks to explore what incentivizes employees to share knowledge in intra-firm CoPs. Design/methodology/approach – The paper presents a longitudinal case study in a large automotive company that introduced 82 cross-functional CoPs into its engineering department. Using extensive qualitative data, two sets of communities: best and worst performing were analyzed.

Findings – It was found that perceived benefits and the employees' willingness to invest individual efforts into community work are stronger in better performing communities. Members of the better performing CoPs drew most benefits from participating in organizational decision processes, as they were able to influence the agenda and create relevant standards. The patterns observed relate to the efforts, benefits, and barriers of community work.

Research limitations/implications – The single case study design limits the generalizability of the results beyond the company studied. Furthermore, some of the data employed were perceptional and relied partly on self-reporting of the community members.

Practical implications – The paper argues that management support for CoPs should aim at influencing the individual cost-benefit calculus of community members. Respecting and implementing results from the communities' work is likely to provide the very basis for innovations to emerge at all. Originality/value – Other than extant studies on CoP performance that focus on company benefits from deploying CoPs, this paper offers a new perspective by exploring the benefits and incentives available to community members.

Keywords Knowledge management, Performance management

INTRODUCTION

Motivating employees to participate in intra-firm knowledge sharing activities is notoriously difficult because it depends on their willingness to voluntarily share their experiences and insights (von Krogh et al., 2000; Wenger, 2000; Kogut and Zander, 1996; von Hippel, 1998; Hildreth and Kimble, 2000). Top management faces the challenge that it can allocate employees to knowledge sharing activities, it can assign them to dedicated projects and communities of practice and order them to contribute their knowledge to the organization, but valuable contributions can not be forced (Borzillo, 2009). Remedies suggested in the literature include bonuses tied to knowledge contributions (Huber, 1991), care (von Krogh, 1998), a sequential process detailing every step of knowledge transfer (Szulanski, 2000), informal socializing opportunities (Nonaka, 1994), team building activities (Huber, 1991), or, notably, the establishment of communities of practice (Brown and Duguid, 2001; Lesser and Storck, 2001).

Communities of practice have been found to support knowledge creation by enabling knowledge sharing among experts in firms (Kodama, 2007; Brown and Duguid, 2001; Wenger, 2000; von Krogh, 2003; for a critical review see Amin and Roberts, 2008). They are "groups of people informally bound together by shared experience and passion for a joint enterprise" (Wenger and Snyder, 2000, p. 139). Communities of practice exhibit three fundamental charcteristics which Wenger (2004) describes as domain, community and practice. The domain is the area of knowledge which community members which to explore. The community is formed by the people to whom this knowledge domain is relevant and who interact and develop relationships enabling them to address issues and share knowledge. The practice is defined as "the body of knowledge, methods, tools, stories, cases, documents which the members share and develop together." (Wenger, 2004, p. 3).Communities of practice substantially differ from formal teams concerning purpose, membership, commitment and lifespan: they develop around a common topic domain that is self defined instead of preset by management, membership in communities is self-selected instead of assigned, commitment arises from a common passion for a topic instead of formal job requirements or project objectives and the lifespan of communities depends from the interest of the members in the topic rather than form reorganizations or project completion (Wenger and Snyder, 2000, p. 142). Communities of practice provide the organization with an enabling context that allows for individual and group learning in knowledge sharing processes (Lave, 1988; Wenger, 1999; Wenger and Snyder, 2000; Lave and Wenger, 1991). They enable the improvement and transformation of the practices they are centred on (Berends et al., 2007) because they constitute the perfect context for informal knowledge sharing among experts: Anderson et al. (2001) demonstrate in their case study that when seeking for information, aerospace engineers prefer informal communication with specialists who have closely related interests.

However, the question remains what incentivizes employees to share their experiences in intra-firm communities of practice. To answer this question, the authors conducted a longitudinal case study in a large automotive company that introduced cross-functional communities of practice into its engineering department. A total of 82 communities of practice were examined over a period of two years. It was found that, the organizational context being equal, some communities performed significantly better than others. The literature on communities of practice and organizational learning did not help us distinguish between performance differences across communities as it highlights the voluntary aspirations of community members to learn together for solving problems related to own work as the driving force for engagement in the community (Wenger and Snyder, 2000; Thompson, 2005; Brown and Duguid, 1991). However, all communities in this sample seemed to share this aspect as all members joined voluntarily and yet their performance varied. The literature on communities of practice performance tends to skim over intra-organizational differences, such as the benefits and incentives available to community members, in favor of a perspective on company benefits from deploying communities of practice (Cross et al., 2006; Wenger et al., 2002; Lesser and Storck, 2001; Wenger and Snyder, 2000; Brown and Duguid, 1991). Performance evaluation forms from the best performing 20 percent and the

worst performing 20 percent of all 82 communities were coded. It appeared that community members' willingness to invest individual efforts relates to the perceived benefits and to community performance.

This article proceeds as follows: first, the extant literature which takes up the issues of community performance and participation incentives is introduced and discussed. Next, the empirical setting and research design is described. Subsequently, the findings are presented and, lastly, the findings in light of the theory and future research and draw implications for management practice are discussed.

COMMUNITIES OF PRACTICE PERFORMANCE AND PARTICIPATION INCENTIVES

Knowledge created in intra-firm communities of practice will only take effect at the organizational level if it is "brought to bear on the business practices" (Spender, 2006, p. 238). Engineers invest with their activities into the performance and future of the organization. What are the incentives to contribute to the organizational knowledge pool?

This question is not trivial; it becomes especially relevant when examining the barriers to knowledge sharing in organizations. Barriers identified in the literature include the fear of criticism, rejection or retribution from colleagues in upper echelons of the hierarchy or a culture of suspicion and distrust (Michailova and Husted, 2003; Ardichivili et al., 2003). Knowledge sharing has been found to depend on the relative power and status of the sharing parties (Huber, 1982), the workloads (Huber, 1982) and the rewards and penalties connected to sharing (Huber, 1991), both individual and collective (Cabrera and Cabrera, 2002). In laboratory settings, the mutual sharing of knowledge proved a highly fragile outcome when subjects were faced with conflicts of interest (Gächter et al., 2010).

Managers are said to need to supervise and facilitate best practice development and sharing within CoPs (Borzillo, 2009) and to cultivate a context of trust and care that induces colleagues to share their knowledge and feel at ease when voicing their opinions (von Krogh et al., 2000). Often highlighted in the literature is the fear of exploitation (Wolf and Wunram, 2003): externalized and shared knowledge creates a "public good" within the organization and reduces one's individual power base. Research on knowledge sharing also uncovered potential cognitive barriers connected to reciprocity and other cultural norms (Barachini, 2009), organizational subgroup affiliations (Rangachari, 2009), as well as shared language and vision (Chiu et al., 2006).

Communities of practice seem to alleviate many of the above-mentioned barriers to knowledge sharing. However, not all communities of practice perform equally well. All other things equal, some perform better than others and members of one community contribute more than members of other communities. Existing research is up to date not able to explain this phenomenon totally. So far, the aim in research was to monitor benefits and to justify organizational investments in communities (Wenger et al., 2002). Consequently, studies investigating community performance focus on revealing what benefits organizations gain from community work (Lesser and Storck, 2001; Wenger and Snyder, 2000; Brown and Duguid, 1991; Berends et al., 2007; Wenger et al., 2002). Scholars did not investigate yet what incentivizes employees to contribute to knowledge sharing in some communities more than in others. With the study reported in this article, the authors aim at filling this research GAP. Below, the case study setting and then the research design and the methods applied are described in more detail.

Research setting

The division "Passenger car development" of a large automotive company decided in early 2000 to set up communities of practice in eleven technology fields in order to foster horizontal collaboration on passenger car elements and processes across model ranges. At the same time, a searchable know-how database should be created, aimed at supporting documentation of key experiences and lessons learned.

The process started with the set up of a project team which was assigned the task to develop and implement this database and to set up the cross-functional communities of practice for the entire organization within two years. These CoPs were conceptualized as intraorganizational expert groups. The division "Passenger car development" was located at two sites which were located 30 kilometers away from each other. However, model range departments working on similar car elements were using the same technical infrastructure and therefore located at the same location.

One of the authors of this article was member of this project team and responsible for monitoring the project's success, i.e. the implementation of the CoPs into the department. The management board expected the communities of practice to meet up once a month for four hours and discuss cross-functional issues as well as to document commonly developed solutions. Apart from working for the company, this author was using the case study as empirical basis for her PhD thesis (Wolf, 2004) where she investigated changes of perceptions as well as patterns of sense making on the deployment process by different process stakeholders over time. She therefore kept track of the major developments during the community of practice deployment process in a work-focussed observation diary (Webb, 2009). The authors are thus able to provide below a short but comprehensive overview on this process based on participant observation (Guba and Lincoln, 1981; Patton, 2002).

The course of the community of practice deployment project can be divided into four distinctive phases:

1. the conception;
2. implementation;
3. improvement; and
4. reflection.

Phases	*Duration*	*Main actors*	*Major activities*
Conception	Months 1 to 4 of the project	Project team members	Identification of CoP topics and collaboration needs Development of database prototype Development of an implementation action plan
Implementation	Months 5 to 12 of the project	Project team and CoP members	CoP formation, CoP kick-off and database launch and CoP member training Conduction of first round of performance measurement
		Project members and management board	Development of performance measurement concept Initiation and organization of celebration event
Improvement	Months 13 to 24 of the project	Project team and CoP members	Conduct of an assessment audit in all 82 CoPs
		Management board	Project approval and change of its organizational status into a permanent group within the knowledge management department
Reflection	Months 1 to 12 after the end of the project	Management board	Assignment of official authorization to issue guidelines that reached across model ranges to CoPs
		Project team and CoP members	Development of and engagement into a six-monthly reflection process of CoP work

Table 21: Phases of the project

Table 18 provides an overview on duration, main actors and major activities of these phases.

Below, the different phases are described more into detail as it seems important to provide the reader with an overview on what happened in the CoP deployment process.

The conception phase

During the conception phase (first four months of the project), the project team developed the basic concept for the future communities of practice structure. Project team members reviewed organizational structure diagrams and interviewed

experts in order to identify topics and needs for collaboration across model ranges. From this, they came up with a map of critical knowledge domains indicating relevant community of practice topic areas. Furthermore, a first prototype of the know-how database and an implementation action plan were developed. At the end of this phase, the management board approved the implementation action plan but assigned the project team at the same time the task to develop a systematic performance measurement approach ensuring management information during the project.

The implementation phase

The implementation phase lasted from months 5 to 12 of the project. During this phase, 82 communities of practice and the know-how database were launched. Community building happened as follows: Project team members asked topic matter experts whether they would like to take over the facilitator role for a community in a certain topic domain. Future community facilitators then contacted further topic matter experts across the car platforms and invited them to join the community. For both roles, the decision to join the community was totally voluntarily; there were no sanctions in case of a negative answer. Becoming a CoP member was even not supported by line management as there was no time budget dedicated to the community work so that engagement would capitalize resources of line managers. However, at the end of this phase approximately 1,000 employees out of 5,000 committed themselves to the community work.

After the communities of practice have been kicked off, the project team conducted kick-off events, trainings for the community of practice members as well as developed a communication and a performance measurement concept. During trainings, CoP coordinators were introduced into and coached in their role and CoP members were teached how to use the knowledge base and how to write up contributions for it. The performance measurement concept reflected the claim of the management board for the collection of statistical data such as the frequency of meetings, the attendance rate of the community of practice members, the number of published and planned contributions for the database as well as the accesses rate of documents at the know-how database. The community facilitators were very indignant with the performance measurement concept. They argued that the indicators used allowed no conclusions on the quality of the work achieved. Furthermore, the form of inherent control in the performance measurement activities was perceived as counter-productive. The management board ignored objections by community members and asked the project team to collect the statistical data. Not surprisingly, the analysis showed that all CoPs complied with the expectations formulated by the management board beforehand: All reported three CoP meetings with an average attendance rate of 80 percent and three contributions published in the database. If necessary, one large chapter was divided into three smaller ones in order to meet the requirements.

In month 12, the deployment of communities of practice was deemed complete. A big event celebrated the success adequately. The community facilitators' spirit during the reflection workshops, held every six months, was ambivalent. On the one hand they acknowledged that the communities of practice created platforms for openly and constructively discussing crucial questions that across functional boundaries. On the other hand they asked the project team to clarify with the management board whether the CoPs were authorized to make decisions for topics that reach across functional boundaries. Without decision-making competence, cross-functional work did not make much sense for them as the solutions developed were not considered as mandatory for all model ranges.

The improvement phase

The improvement phase (months 13 to 24 of the project) focused on the optimization of the CoP work. In order to gain insights into the current state of the community of practice work, its problems, opportunities and risks, the project team planed and conducted a complex audit: On the basis of a structured questionnaire that focused on nine core areas of the CoP work, expert interviews were conducted in the 82 communities of practice. Prior to the audit at least one community meeting was attended by the project team members for gaining an impression of the activities of the respective community of practice.

The methods section reports in detail on the measures applied in the audit. Here, it is important to mention that the assessment mark assigned to the CoPs was the product of a negotiation process between the CoP and the project

members. The benefits CoP members perceived from their work in the CoPs was additionally documented a form for capturing success stories. The use of this form in the audits helped the community of practice members to reflect on the value of their work and created enthusiasm and, at the same time, the stories collected were used as marketing instrument towards the management board. The review changed the relationship between the project team and the community of practice members dramatically. The latter perceived the audit as a sensible and supporting counsel. The project team advanced from a bogeyman held responsible for unpopular orders of the management board to a counseling unit representing the CoP members' interests towards the management board.

At the end of the improvement phase, the results of the audit were presented to the management board as part of the final project report. The CoP members made very clear that they would need further continuous support. Subsequently, the management board recognized the success of the project and changed its organizational status from a transient project team into a permanent group within the knowledge management department.

The reflection phase

The active marketing of the achievements of the community of practice work in the frame of the review convinced the management board of their value. At the beginning of the reflection phase, they obtained the official authorization to issue guidelines that reached across model ranges. Suddenly a horizontal, cross-functional structure was in a position to make decisions which concerned all model ranges and to assert them against the vertical hierarchy, i.e. functional management. The management board reserved the right to veto any decision. Yet, the change was radical: everyone who was not a member of the respective Community of practice was not able to influence future guidelines. This change did not solve the problem of the missing budgeting for the communities' work but it lead to renewed motivation of the community members.

With the end of the project, a new phase of community of practice evaluation started. As the management board did not ask the new knowledge management department for further regular justifications and the department saw its task mainly in supporting the CoP work, it developed a new instrument aimed at assisting the CoPs in systematic reflection of their work. The new form will be described in more detail in the method section of this article. The Knowledge Management department triggered this reflection process every six months and collected the reflection forms. From this, the department received a fast overview on the actual situation in the community of practice and was able to offer customized support.

RESEARCH DESIGN AND METHODS

The aim of the research reported in this article was to explore what incentivises engineers to share their experiences in intra-firm communities of practice. As this topic has so far not received a lot of research attention, the pursued research strategy was exploratory. A predominantly inductive and qualitative, hybrid approach was applied that suited the research situation in terms of its participants in the researched department, the researchers and the topic in question. The interpretive paradigm guided the actions undertaken within the context of this research (Maxwell, 2005; Miles and Huberman, 1994).

The authors' interest to study this topic emerged from the results of the CoP audit in the improvement phase in which communities and researchers evaluated the performance of their CoP. Audit results (details below) suggested that all other things being equal, some CoPs in this sample performed better than others: Audits rated each community with a maximal score of 100 percent. The 20 percent best performing communities of practice (14 CoPs) ranged between 86.67 percent and 96.19 percent; the 20 percent worst performing communities of practice (14 CoPs) exhibited scores between 50.91 percent and 67.05 percent. The average assessment score differed clearly: the average of the "best" group was 90.45 percent, those of the "worst" group 63.93 percent.

To exclude other variables that might have an influence on community performance, the authors performed an analysis on community member characteristics. The organizational context was similar across all communities: the community composition was quite homogenous, all participants were of the same profession (engineers with a specialisation in passenger car development), had been working for a minimum of three years in the same company, and were about the same age (average age was 42, members age ranked from 35 to 48 years). In addition, the authors were able to exclude structural characteristics as influential on community performance. As described above, all communities have been set up at the same time, they all received the same structural support in terms of training, counselling by the project team, and rooms and facilities for community meetings. It was therefore decided to analyse the qualitative data available with a specific focus on the research question at hand.

Data sources used

For each of the two community groups (i.e. the best and worst performing), the authors created a set of qualitative data that had been gathered with the different evaluation instruments (see below) applied by the project team from the beginning of the implementation phase (month 5) up to month 12 after the end of the project. The data included perspectives of both the project team members and self-evaluations of CoP members during the different phases of the research. This provided an inside and outside view on the communities from their initiation until the end of the first year after the end of the implementation project. Table 19 gives an overview of the data sources used.

The data sources are described in more detail below.

Audit assessment. The CoP audit rated community performance in six core areas identified by the project team as important for the quality of community of practice work:

1. Efficient organization of community work (invitations on meetings, agendas, topic lists, duration of meetings, roles)?

2. Motivation and team composition (participation, collaboration, fluctuation of community of practice members, topics, guests)?

3. Focus of the community work on the right content in terms of strategic relevance for the organization?

4. Interconnectedness (with other communities of practice, with the steering committee, with working groups, with community of practice external experts)?

5. Quality of chapters contributed to the know-how database?

6. Quality of meeting results in terms of achievement of community objectives?

The audit guideline consisted of these six questions that served to score the community performance. For each of them, the actual community work was rated on a LIKERT scale between 1 (very bad) and 5 (excellent). CoPs could reach a maximum of 30 points corresponding to a score

of 100 percent. The audit guideline furthermore asked for the documentation of answers to the questions "What goes well in the CoP work?", "Where would you like to improve or need support by the project team or the management board?" and "What are tips and tricks you would like to suggest to other CoPs?" for each of the six core areas.

Prior to the audit, project team members attended at least one community meeting for gaining an impression of the actual status of the respective community of practice. The community members prepared for the audit in discussing the six topic areas among them and documenting their self-perception. The audits itself were attended by the author-researcher, two members of the project team who had supported this CoP from the very beginning and from a group of three CoP members including the community facilitator. Audit assessments lasted two hours in average.

The audit assessment can be classified as a mix between expert interviews and group discussions. In a first step, the project team members asked the CoP members to tell their self-perception on what was good concerning their work and where they would like to improve or would need some support by the project team or the management board. During this (expert interview) part of the assessment, the present CoP members were treated as experts in their own field of activity, i.e. their CoP work. They represented a group of specific experts, namely the members of their CoP; and the semi-structured interview guideline was used to restrict the interviewees to their area of expertise (Flick, 2009).

Evaluation instrument	*Point in time of application*	*Representing the perspective of*	
		Project members	*Community of practice members*
Audit documentation (28 completed review evaluation forms, 140 typed pages)	Improvement phase (from month 18 to 22 of the project)	After the discussion/negotiation between community of practice and project members agreed assessment of the quality of the community of practice work in the six areas in question: Documentation of "What is going well?" Documentation of "What needs to be improved?" Documentation of tips and tricks for other CoPs	
Form sheet for success stories (48 completed form sheets, one page each)	Implementation, improvement and reflection phase (from month 12 of the project until month 12 after end of the project)	Not applicable	Perceived benefit (qualitative and quantitative) of the community of practice work for both the organisation and the community of practice members
Reflexion sheet (41 completed reflection sheets, one page each)	Reflection phase (month 6 and 12 after the end of the project)	Not applicable	In a reflection session in the community of practice agreed perception of the quality of the community of practice work in the six areas in question: Documentation of "What is going well?" Documentation of "What needs to be improved?"
Observation diary (120 typed pages)	All phases (from month 1 of the project until month 12 after the end of the project)	Documentation of statements of the members of the project on the quality of the community of practice work as well as on problems and challenges in the different communities of practice	Documentation of statements of the community of practice members on the quality of the community of practice work, on benefits as well as on problems and challenges

Table 22: Evaluation instruments used as data sources

After having documented the answers of the CoP members, the members of the project team gave their own impressions and negotiated with the CoP members an assessment mark for each of the six core areas. They documented the negotiated assessment mark as well as the result of the discussion on what was perceived as good and what should be improved. The value of the community work perceived by the community of practice members was additionally asked for and documented in the form for capturing success stories. This part of the assessment resembled much more a group discussion where participants negotiate consensus on a certain issue (Krüger, 1983) and develop strategies for solving problems through the discussion of alternatives (Flick, 2009). The author-researcher took additional

notes on the negotiation process in the group which she later transferred into her research diary and used exclusively for her PhD.

Form sheet for success stories. CoP members have been free to fill in the form sheet for success stories at any time. The form sheet included three questions:

1. What is the concrete incident from your CoP work you would like to report on?

2. What is the qualitative benefit you would like to report?

3. If applicable: What is the quantitative benefit (cost or time savings) you would like to report?

Success stories reported with this instrument represented solely the perspective of the CoP members. Therefore, CoP members have been asked in trainings to support the reported benefits by further documents if possible.

Reflexion form sheet. The reflexion form sheet was the successor of the audit documentation form and the form sheet for success. It additionally asked for statistical data in terms of an indication of the frequency of meetings and the attendance rate of these meetings. Data gathered with this form sheet were self-reported and the result of a discussion among the CoP members.

For understanding how perspective biased the data reported in the form sheet for success stories and the reflection form sheet were, the authors compared them to what has been documented during the audit assessment. It was found that although data indicated a specific CoP perspective and the development of the CoP over time, none of the CoPs reported issues which were very surprising or did totally not fit the picture that was created during the audit. However, it is obvious that the authors are not able to totally exclude relational biases produced by the situation of reporting to a support team in any of the data (Miles and Huberman, 1994).

Observation diary. The author (and member of the project team) who aimed to use the empirical case study for her PhD documented the major developments during the community of practice deployment process in a work-focussed observation diary (Webb, 2009). She documented on a daily basis what happened to her, to whom she had been speaking and what she talked about to these people for keeping track on who or what influenced her perceptions during different times of the process. Documented items took the form of descriptions of situations and events or quotes of utterances of other people and subjective interpretations. The data collected with the diary are valuable in the sense that they provide insights into the subjective perceptions of the author-researcher, but limited as they involve a selection bias (Alaszewski, 2006).

Data analysis

The qualitative data in the two data sets (i.e. the best and worst performing CoPs) which served as our units of analysis took the form of statements by project team members and community members on both good and bad aspects that influenced the performance of the community work.

For data coding, the authors applied Miles and Huberman's (1994) thematic coding approach for generating meaning out of a large amount of qualitative data. This coding approach involved three steps: At first, "first-level" codes (Miles and Huberman, 1994, p. 69) taking the form of categories or metaphors were assigned to the data for labelling units of meaning. Thereafter, "first-level" codes were grouped together to "pattern codes" which represent emergent topics (Miles and Huberman (1994, p. 69). Finally, codes were mapped and relations between them were noted and displayed as components in a network of themes (Miles and Huberman, 1994, pp. 70-71). The software Atlas.ti was used for coding. The coding strategy was not exhaustive, i.e. not all data fit into a "first-level" code.

The authors applied an iterative coding procedure involving multiple interpreters for ensuring credibility of the data analysis (Patton, 2002): In a first step, the data was coded by the author who was a member of the project team. This author knew the empirical context from personal experience and was able to interpret and contextualize statements by both the community of practice members and the project team members adequately. In a next step, the other two authors checked the validity of the coding. As they did not know the context in which the statements had been made, they came up with additional codes, interpreted certain statements differently and sought clarification. The discussions of possible interpretations lead to a subsequent change and refinement of the codes. Thereby, the findings were iteratively improved until all authors agreed on the codes and the thematic maps developed. Interpretation of qualitative data in groups is an important step to reduce the influence of the frame of reference of one researcher and to increase the validity of the interpretation (Steinke, 2004).

RESULTS

During the coding process, the authors linked 1096 text passages (662 in the "best" sample, and 434 in the "worst" sample) to a total of 48 codes. From the coding, three thematic clusters evolved: CoP members talked about:

1. the efforts they invested or did not invest into the community work;

2. the benefits they perceived from participating in to the community;

3. the barriers they perceived in their community work.

The quantities of the text passages for each code from the two data samples are displayed in Tables 20-22 respectively.

This section reports results of the three thematic clusters, namely first the effort investments community members made into the community work and which indicate the extend to which CoP members were motivated to participate and contribute; second benefits they perceived from the community work and which display reasons community work and which display reasons why they participated and contributed; and third barriers that accompanied the collaborative work in the communities. It seeks to identify if and how they differed across the two data sets of communities of practice.

Efforts invested

Table 20 represents the codes related to the efforts that community members were ready to invest into the CoP work.

A main trend that emerged was that members were willing to invest more efforts in terms of time and contributions in best performing communities (130 mentions) than in the least performing (51 mentions). One CoP coordinator for example explained:

> *I have included the time which is necessary for Knowledge Management work in my annual plan during my appraisal.*

The efforts community members were not ready to invest present an inverted picture. The authors found only 90 instances where members of best performing communities of practice stated that they were not willing to invest a certain effort, while 164 examples in worst performing communities where members said that they would not take on specific issues could be identified. The following statement of a CoP coordinator from the worst CoP sample exemplifies how these statements looked like:

> *To our members, the operative tasks of the day-to day business have a higher priority than the work on contributions to the knowledge data base; Knowledge Management does not have such a high priority.*

Table 23: Efforts invested and not invested

Topic code	*Efforts community of practice members do invest* "Best"	"Worst"	*Efforts community of practice members do NOT invest* "Best"	"Worst"
Community of practice coordination (planning, conduction and post processing of Community of practice meetings and other community of practice activities)	42	17	3	5
Contributing to planning and write-up of articles for the know-how database	29	17	38	54
Coordination of activities with other communities of practice	21	7	3	15
Actively engaging in community of practice meetings with presentations, own contributions	16	3	16	28
Participating in community of practice meetings	14	6	8	22
Reflection of community of practice activities, organisation of community of practice strategy meetings (off site)	5	1	5	27
Setting up and updating the community of practice intranet page	3	0	8	11
Transfer of community of practice results into vertical (line) activities	–	–	9	2
	Σ = 130	Σ = 51	Σ = 90	Σ = 164

Note: Numbers in cells represent the absolute number of mentions of the topic in question in the data

In the best performing communities in this sample, members were willing to invest substantially more efforts. Statements from members of the latter communities of practice indicate that attendance rates were high, one CoP coordinator for example highlights the "more than 86% attendance rate!". Furthermore, these CoPs held additional "regular knowledge exchange with the communities X and Y" to facilitate exchange.

Codes: Perceived benefits (PB)	*"Best"*	*"Worst"*
General comment: members perceive a benefit of community of practice participation	19	2
PB resulting from participation into decision processes	109	16
Participating in and impacting development and definition of best practices	41	3
Impacting standardisation in vertical (line) processes and across platforms	26	2
Contribution to innovation management processes (positioning/bringing in community of practice ideas, participation in selection processes for new product development)	23	3
Participation in the development of a common strategy for dealing with and assessing of suppliers	16	4
Impacting strategy development processes	3	4
PB resulting from building up of relationships/networks/reputation	68	32
Build up of networks/connections/relationships with competent people (other experts) from different car platforms	59	31
Close contact and access to senior management across car platforms	9	1
PB resulting from learning together/knowledge sharing in social learning processes	122	54
Gaining information on hot (up to date, interesting, innovative) topics and discussing them with other experts	43	13
Open sharing of experiences with other experts across car platforms	35	22
Build up of own professional knowledge on technologies and methods	13	10
Impacting development and learning a common language/terminology across car platforms	12	0
Participating into the conduction of technology benchmarks/discussion of technology benchmark results with other experts	9	6
Transcontinental sharing of experiences with other experts	8	0
Participating in the conduction of market and client analyses/discussion of market and client analyses results with other experts	2	3
	Σ = 318	Σ = 101

Note: Numbers in cells represent the absolute number of mentions of the topic in question in the data

Table 24: Perceived benefits of community of practice work.

Members in the worst performing communities were not willing to invest efforts in many activities, including the participation in meetings and the planning of activities within the community. For example, one CoP coordinator of the worst performing CoPs states: "We still did not establish long term planning of community meetings". Coordination with other communities was not systematically managed, as this statement of a CoP coordinator exemplifies: "We should intensify networking and ensure continuous participation in meetings of other communities". Similarly, critical self-reflection of activities did not happen regularly according to the CoP coordinators: "The reflection on benefits from community participation of the members did not take place due to time limitations.". Once, for example, a community even declined to meet a member of the observing project team for a review session.

Statements of members in worst performing communities also indicated that attendance rates in meetings were often below 50 percent and important people were missing as this CoP coordinator explains: "Directly affected members do often not participate – participation rate of only 46%". In summary, it can be observed that the contributions differed widely across the two sets of communities. Members did contribute more to better performing communities.

Benefits perceived

Did members who contributed more also receive more back in return? Table 21 presents the resulting codes related to benefits that community members perceived from contributing to the community and reports on the frequency of the code occurrences in both data sets.

The 15 codes that emerged during the analysis have been subsumed under three main topics:

1. participation in decision processes;

2. building up of relationships/networks; and

3. learning effects/knowledge exchange in social learning processes.

In general, community of practice members from the "best" communities mentioned beneficial aspects of their work three times more often than community of practice members from the "worst" communities (318 text passages versus 104 text passages), indicating that these members perceived more benefit from the community work.

The most significant differences can be seen in the first category: members of best performing

communities of practice participated much more into and impacted organisational decision processes (109 versus 16 instances mentioned). One CoP member summarizes for example the benefits he experienced in the following statement: "The community is integrated into decision processes and contributes standards."

Being able to influence strategy development, contributing to the selection process of ideas for future innovation management projects as well participating in the creation of organization-wide best-practices and standards was perceived as beneficial by contributing members. Members of best performing communities of practice were able to decide on a set of standards within their area: "We develop and define best practices and specifications.". Contributing members thus brought up topics and set the agenda of upcoming standards and best-practices. Members of one community experienced for instance an "improvement of the product maturity stage through agreed standards." Members who did not participate or contributed less had less control over what would become organization-wide practices.

Also, in both other code categories the differences were obvious. Analysing the text passages from the "worst" communities of practice sample, it can be seen that members did profit from networking, however, much less so than in successful communities (32 versus 68). Being able to get access to and benefit from relationships mainly across functional domains and to top management proved to be a benefit accessible to contributing members. Statements from members of communities of practice stated that especially being able to work with experts from other car platforms was beneficial: "In community meetings, experts from different domains meet and develop solutions of concrete tasks." On contrary, in the data from the worst CoPs we find statements like this one: "Due to the heterogeneous composition of the CoP, the work on relevant questions is difficult."

Personal learning from the exchange between experts appeared to be the third benefit category that has been identified. Members of the best performing communities of practice mentioned it more than two times more than members of the worst performing ones (122 versus 54 mentions). Contributors were able to develop and learn a common language and terminology across geographical and divisional boundaries: "A first benefit has been reached through the standardization of the terms used." They were informed about current "hot" topics in those discussions, and learned from openly shared experiences of experts in other fields: "The community is a very good information platform concerning all relevant topics in the topic area." Those who shared also received feedback and advice from others resulting in direct benefits for their day-to-day work. Members of the worst performing communities, on the other hand, did not think highly of the discussed topics: "Actual topics are discussed only to a very small degree." Only 13 times, topics were classified as attractive or interesting, compared to 43 times in the best performing communities.

When looking at the set of "worst" communities of practice only, another pattern emerges. The two most frequently cited benefits were open sharing and the networks and relationships to other competent individuals. While these benefits were mentioned far less frequently than in the set of "best" communities, they allow the conclusion that community members benefited from networking even in the context of low performance communities of practice.

The benefits available through engagement in the community led to increased commitment (which was not perceived as costly) and increased perceived benefits and, ultimately, better community performance. (see Table 20)

Barriers to CoP work perceived

The third thematic cluster reported on barriers CoP members perceived to their community of practice work. Here, the authors examined whether performance differences arose because the best performing communities faced fewer barriers. No attempt was made to classify the task complexity of the specific communities and all statements were coded according to the perceived difficulties and hurdles. Table 22 summarises the statements related barriers in the communities' work.

Although community of practice members from the "best" and the "worst" sample mentioned almost the same amount of hurdles, they emphasized different issues. The emerging codes were subsumed under three categories:

1. structural barriers;

2. unclear goals, power and responsibilities; and
3. community work/topic related issues.

Codes: Perceived barriers to community of practice work	*"Best"*	*"Worst"*
Structural barriers	36	22
There is no time budget dedicated to the community of practice work	18	19
Content on which the community of practice works is secret and cannot be published openly in the know-how database	10	1
Not all community of practice members have access to the know-how database	8	2
Barriers resulting from conflicts with line management and unclear responsibilities	38	20
Community of practice members do not have the authority for making decisions	19	15
Community of practice activities are controlled by senior management (checklist, management reports etc.)	14	4
It is not possible to measure the benefit of the community of practice work quantitatively (time and cost saved)	5	1
Community of practice work-related barriers	1	32
Topics discussed during community of practice meetings are not interesting/attractive	0	16
Discussion during the community of practice meetings is not results-oriented and does not lead to decisions	0	11
Members do not perceive a benefit from community of practice participation	1	5
	$\Sigma = 75$	$\Sigma = 74$

Note: Numbers in cells represent the absolute number of mentions of the topic in question in the data

Table 25: Perceived barriers to community of practice work.

While the lack of a time budget for community work was perceived as equally limiting, members of the best performing communities of practice were much more concerned about the lack of unrestricted access to the resulting documents: "Access authorization for community members from plant X is dissatisfactory". A further problem mentioned by members of the worst CoPs was documentation restrictions due to the confidential state of some information: "Documentation is limited due to the confidentiality of the topic.". These issues did not seem to be a problem or were at least not perceived as one in the lower performing communities.

Second, best performing communities were more sceptical about being controlled and monitored. They demanded more authority and less monitoring of their activities like this CoP coordinator: "The status form is not purposeful, the people feel imposed!" Filling in status forms has been perceived as "stolen" precious time in which they could have been productive. Third, almost exclusively the communities of practice in the "worst" set perceived significant work-related barriers. The topics discussed were deemed unattractive and led to no relevant decisions, like this statement exemplifies: "The presentation and common discussion of technologies relevant for our area has not happened yet." Altogether, there was a lack of benefits for community members. The best performing communities of practice seem to suffer – or perceive to suffer – more from external barriers while the worst performing communities of practice have a much more inside focused perception. For instance, only members of the worst performing communities lamented about the "lack of coffee and cake at meetings".

In summary, none of both sets of communities enjoyed a better position from the beginning or gained more financial support than others. The communities of practice from the "best" set claimed more frequently that they faced external barriers, whereas the communities of practice from the "worst" set would complain more about internal work-related issues and found discussions fruitless. In addition, members' perceptions of the costs and benefits displayed clear differences between the two sets of communities studied.

DISCUSSION AND IMPLICATION

This study of 82 communities of practice in the engineering department of a large automotive firm revealed that the communities differ greatly in performance. The employees' willingness to invest individual efforts into community work was shown to relate to the perceived benefits and to community performance. A community that created personal networks, fostered discussions of relevant topics and created standards of practice was perceived by its members as worth participating. Vice versa, a community without engaged participants led to less perceived benefits. Members of the lower performing communities were not willing to invest in coordination and reflection activities and perceived the efforts to exceed the benefits. The nature of this "chicken and egg" relationship is precisely the point the authors intent to make. Perceived benefits grow from collaboration and simultaneously increases collaboration as the perceived benefits become stronger.

While the initial emergence of perception of benefits from community work cannot be traced nor explained in this study, the authors observed clear differences in the availability of benefits across communities. The differences gave rise to observable patterns that allowed us to cluster and characterize them. Three types of perceived benefits could be indentified:

1. learning effects/knowledge exchange in social learning processes;

2. participation in decision processes; and

3. building up of relationships/networks.

As Wenger (1999) and others observed, communities of practice foster relationships and social learning first of all. This study confirms this result but find that benefits pertaining to relationships and networks, while useful, only represent one type of benefit. Second, members of the best performing communities of practice drew most benefits from participating in organizational decision processes, as they were able to influence the agenda and create relevant standards. Third, the establishment of new networks and personal relationships to other competent individuals inside the firm characterized best performing communities of practice vis-a`-vis less performing communities.

A few limitations apply to this research: The single case study design limits the generalizability of the results beyond the company studied. Other industries, the company size, and the communities' location within (or across) the organization could impact on the members' willingness and cost-benefit calculus to contribute to the activities of a community of practice. The approach chosen to capture members' perceived costs and benefits might have missed important cognitive barriers or facilitators of knowledge sharing (Barachini, 2009). However, some of these potential barriers are limited by the homogeneity of this sample of communities that by and large operated within one cultural environment.

Furthermore, some of the data employed was perceptional and relied partly on the self-reporting of the community members. The perceptions of benefits and efforts are inherently subjective (and inter-subjective) and may change over time. One author accompanied the communities and their evaluation sessions over the time period studied to ensure the consistency of the reports that underlie the analysis. This limitation holds implications for the potential use of information support systems that could be used to circumvent the role of the mediator when assessing community of practice performance. More applied research is needed here.

Three implications for future research stand out. First, it remains unclear how the creation and the benefits of community contributions are linked in time and how the benefits become visible to new joiners. If the emergence of perceived benefits is tied to collaboration, what determines if a virtuous cycle creates perceived benefits or if a vicious cycle deters engagement and increases perceived costs? The understanding of these dynamics appears central to the management of communities of practice and other forms of collaboration and research only started to shed light on this issue (Spaeth et al., 2008).

Second, the characteristics of perceived benefits identified in this research could inform organization theory and practitioners about the nature of incentives relevant for knowledge sharing. If individuals perceive networking opportunities and agenda setting (and others) as key benefits that induce them to share knowledge, this insight could translate to employment contracts, office layouts, fringe benefits, as well as work routines and structures that better accommodate for knowledge workers' needs. While the effect of learning and relationship building are well under-

stood, the participation in decision processes may have been overlooked as a motivating factor in community of practice work.

In particular, the translation from innovations generated in communities (standards, best practices) to the organizational level is well understood by community members and anticipated in their willingness to share their experiences. Applied research in communities of practice may test whether transparent decision and translation processes will in fact boost perceived benefits of community work.

Third, the level of formality of communities of practice may play a less important role than the literature suggests (Wenger and Snyder, 2000). This study examines variance in performance across a large sample of communities of practice that have been established formally yet with the usual characteristics of voluntarism and hierarchical independence. It remains unclear whether the formal establishment of communities of practice has any impact on their performance and future research should compare formal with informal communities to control for this potential effect. Special attention should be given to internal communication and how voluntarism is understood in each case: communication and framing might impact perceived costs of contribution.

The results hold implications for management practice. Recognizing that communities of practice may help to solve the collective action problem of knowledge sharing in firms may lead to action in two areas. Support for innovation can start at the individual or at the community level. While the literature has adequately described the support for communities of practice, the individual cost-benefit calculus can also be favorably influenced. Lower perceived costs of contribution by very engaged members led to increased commitment.

The established insight that communities of practice contribute to knowledge creation by encouraging knowledge sharing in an organization deserves refinement. While top management should foster communities of practice (Brown and Duguid, 1991; Wenger and Snyder, 2000; Borzillo, 2009), the way in which communities of practice are tied into the organization is likely to exert direct impact on benefits perceived by community members. The latter derive benefit from knowing that their contributions translate into new knowledge and most probably innovations at the organizational level.

Members of high-performing communities of practice in the sample reported personal benefits deriving from the contributions to organizational standard setting and best practice development. Respecting and implementing results from the communities' work is likely to not only contribute to innovation in the organization but to provide the very basis for the innovations to emerge at all. It seems that only if the innovative ideas and approaches of CoP members translate to organizational practice, that is only if the knowledge from the community is transformed into organizational knowledge (Spender, 2006; Luhmann, 2000), community members will perceive benefit from their contributions to the collective work in communities.

5. Conclusion and issues for future research

This thesis defined collaborative innovation as a collective development process of new and useful products and services across and outside firm boundaries. From a strategy perspective, the essays gathered in this thesis established key activities when entering new markets with products based on collaborative innovation and, even earlier, our findings call for a cautious setup of initial conditions of sharing that appears as highly fragile in the laboratory. In a nascent area of the entertainment industry, we identified a group of companies commercializing innovations related to the new genre in film called Machinima: the shooting of film inside video games. These companies innovated with technology they co-developed with peers outside their firms. They developed production technology and evolved skills pertaining to Machinima production that allowed for their entertainment products to sustain an audience. At the same time these firms recognized that the user innovations they developed was not enough to succeed in the motion picture industry. They hired experts in cinematography to bring in and teach them how to produce better films. An implication from this finding is that strategic collaborations between firms and individuals or between firms and user communities can be instrumental to succeed.

In order to learn more about the very initial conditions of collaboration as needed for collaborative innovation, as defined here, we took a simplified setting to the laboratory. Remember that our definition involves collaboration across firm boundaries. This implies that hierarchies and contractual obligations may not (yet) bind collaborators and the initial conditions may be subject to strong conflicts of interest between the collaborators. A question of strategic interest was whether and how the initial collaboration can sustain. Our laboratory study revealed that the conflicts of interest of the follower, the collaborator who picks up a suggested collaboration or the "second" of two collaborators in the simple model, impact collaboration more adversely than the conflicts of interest of the first collaborator. This implies that business and government policies to protect initially shared content from potential appropriation (and limit the opportunity costs for the follower) can have significant impact on sustaining collaborative innovation.

A third topic in strategy concerned the use and potential benefit of social software. Social software, by definition, allows groups of individuals to interact. The groups may span organizational departments and teams or entire organizations. The interactions may enable collaborative innovation and even entrepreneurship (Hienerth, Keinz, and Lettl, 2011) or lead to stifling political processes and demotivation within organizations (Denyer, Parry, and Flowers, 2011). When deploying social software designed to facilitate collaborative innovation it is strategically important to accompany its use and prepare for ambiguous reception and changing behavior among potential collaborators. While this thesis points to the myriad of promises of collaborative innovation we also consider the possibility of flawed incentives and the uncertainty inherent in introducing tools all too frequently heralded as automating collaborative innovation.

From the perspective of technology, collaborative innovation appears as straightforward at first. Open systems allow for reuse, thus developers will reuse extensively if reuse is easy and free. However, even within organizations developing software, code reuse is notoriously difficult to implement on a large scale (Kim and Stohr, 1998). Between firms and communities, relationships can be fraught with lack of trust or the suspicion that the other may not adhere to own standards of quality or reciprocation (Faraj, et al., 2011; Shah, 2006; Dahlander, 2007). Our work documents reuse on the level of code across open source software development communities and suggests policies and strategies on the level of system development taking into account the possibilitiy of reuse and joint development. It becomes clear that reuse decisions as the very basis of collaborative innovation processes touch upon the behavior of various actors beyond "the user developer" and "the firm develper": In fact, management on several levels of hierarchy in the firm are implicated when it comes to initiating and sustaining technology reuse and joint development strategies with parties outside the firm. Critical research questions going forward include the work on frameworks for establishing an open design methodology that takes

into account repeated, reliable, and measurable processes of joint development that tie firms and communities together in their quest for building better technologies. Developing code or entire systems with user communities may imply contributions to a public good and clear managerial goals and policies about how the sharing of internally developed knowledge with a public proceeds and how (Jarvenpaa and Mazchrjak, 2010). Table 23 gathers nine select issues for future research that stand out from the topics covered in this thesis in addition to each of the essays in this collection drawing conclusions and outlining issues for future research with respect to the specific contribution made in the essay.

Collaborative innovation	Working papers	Select research issues
Strategy How to enable collaborative strategy?	Firm-community interaction as a dynamic capability (with Evila Piva and Cristina Rossi Lamastra) Reconfiguration as the third dynamic capability in collaborating with stakeholders (with Fotini Pachidou and Georg von Krogh	• Strategic collaborations between firms and (online) communities • Characterization of capabilities pertaining to external resource acquisition and integreation • Creation and appropriation of value from collaborative innovations
Technology What technologies are developed and reused?	Role of component dependencies in the development of complex systems (with von Krogh, Stuermer, Geipel, Spaeth, Baldwin) A model of lightweight component reuse in OSSD (with Spaeth, von Krogh, Stuermer)	• Knowledge and technology transfers across communities and firms • Open design: technology integration in development • Private-collective innovation by firms
Social practice Who collaborates and why?	The competition between institutions and social practices in the consumption of pharmaceuticals (with Pachidou and von Krogh) Mavericks of Machinima: the creation of a new genre involving directors and audience (with Stefan Meisiek and Georg von Krogh)	• Changing social practices as threat and promise to firms and institutions • Knowledge creation processes as sociomaterial practices • Ba as the space and time for relating to others in collaborative innovation

Table 26: Issues for future research on collaborative innovation.

The secion on soclial practice starts with our review and theory article on motivation of open source software developers. While it offers a complementary perspective on individual motivation adding to the established lens of self-determination theory, its main contribution lies in framing institutional change and a drive for quality in technology development as issuing from the social practice that developers come to share and evolve. In that essay, the notion of social practice derives from the work of Alasdair MacIntyre and resonates a deep concern for ethical aspects in practice. It is based on socialized and internalized understandings of good and right ways of doing things, developing software in our case, that individuals come to appreciate, embody, and demand of others an adherence to standards of excellence that become definitive of the social practice they share. The importance and impact of these shared values should not be underestimated and we theorize that institutional change may follow changes in values in the social practice where institutions are no longer thought fit for supporting the social practice. In other words, an institution, such as a firm or a regulataion, may support collaborative innovation and the social practice that carries out the process of collaborative innovation. If that institution, however, no longer complies with the values that define the social practice, it may be rejected or reformed to better fit shared values. We suggest that these mechanisms are not yet well understood but may prove critical for collaborations between firms and user communities.

Rather than delving deeper into the specific issues for future research outlined by the essays above, I wish to open up a few lines of thought that broaden the scope for an agenda on collaborative innovation beyond what's been done so far by our group of co-authors. Social practice theory offers an attractive lens with which to approach the dualities of individual and collective, subject and object, knowing and doing, which permeate the literature on knowledge creation theory and seem all too frequently to assume entities rather than the process of creation at the origin of new knowledge. With a focus on action we may be better positioned to theorize knowledge creation at the point where human

and material agency overlap and embodied ideas can emerge into consciousness and be retained (Varela, Thompson and Rosch, 1993; Barad, 2003).

Future research on collaborative innovation should probe deeper into the origins of organizational knowledge creation, the starting point of the spiral, or the nature and conditions for an enabling context or Ba (Nonaka, Toyama, and Hirata, 2008; von Krogh et al., 2000), which is the place and time where socialization between individuals becomes productive and knowledge takes form and can be retained. The origin is not a singular point in time but an organizational routine in need of enactment, maintenance, and structuring (Feldman, 2000 etc.). Hence, we start with the presumption that action lies at the heart of knowledge creation because in the theory not only is knowledge "the capacity to act" (Nonaka and von Krogh, 2009) but also action the capacity to know (Nicolini, 2011).

The interplay of human and material agency in changing technologies and routines has been captured by the metaphor of imbrication (Ciborra, 2006; Leonardi, 2011), which is an architectural metaphor that describes the interdependence of two types of roof tiles to suggest that human and material agencies are closely linked to produce a working routine or technology. In Leonardi's words, "human and material agencies are the shared building blocks of routines and technologies" (2011: 149). Practically, this means that people change office layouts or software code or take shortcuts in order to reach their goals. The physical environment and technologies afford certain freedoms to act or interact and they constrain at the same time. That material and human agencies act jointly is obvious but how they combine is less understood, less even how the entanglement of the social and material environment (Orlikowski, 2010) impacts collaborative innovation.

Insights from architecture may help social scientists better understand how material agency through the physical layout of cities and spaces for work, including digital materiality (Gramazio and Kohler, 2008), influences how people interact. The three questions developed below emerge from a confrontation of material and human agency at the very front end of the innovation process that is organizational knowledge creation: First, how are standards of excellence brought into knowledge creation processes and how are they changed and retained? (consequentiality) Second, how does individual experience appear and change the dynamics of knowledge creation in groups? (intercorporeity) Third, how do sociomaterial practices give rise to routines for organizational knowledge creation processes? (purification).

The spiral in Nonaka's (1994) article on knowledge creation is a brilliant metaphor because it not only teaches to study knowledge creation on the continuum from tacit to explicit knowledge but also in an oscillation between the individual and the collective. A sociomaterial practice perspective helps in approaching both dimensions because both, that is the presumption, reflect a duality that should be studied as one (connectivity or sequentiality) rather than as a separation of poles from the start. The emphasis on the start of the spiral is crucial here because it brings to the fore the inevitable theoretical conundrum of the prior, the boundary object and material, embodied context in which ideation takes place. Retained technology and routines become artifacts in collective use and thus entangled in every further oscillation. Conceptually, the starting point is an intention, a perception, and an experience: in other words, how does individual experience appear and change the dynamics of collaborative innovation?

Individuals intuit and experience new ideas and insights (Hodgkinson et al., 2009) that they value as something worth exploring further (Spinosa et al., 1997). While ideas may appear as strokes of genius, organizational knowledge creation theory is concerned with group work and socialization as the practice of ideation and development (Nonaka and von Krogh, 2009). The experience of the collective expresses in action towards and with others: it is here that future research may aim at contributing by exploring the relationship between individual experience and collaborative innovation processes. The work by Monteiro (2010) uses visualization to show how groups interact with digital objects to create scientific insights.

More practically, future work on collaborative innovation may ask how individual experience appears and changes the knowledge creation process when it occurs in the context of collective work. Researchers may study the patterns of how individuals experience knowledge creation processes and how process dynamics relate to individual perception and relations to each other in a state of spontaneity and flow. A contribution could lie in a more precise concept of what Ba means to the individual and

collective in terms of experience and creativity when relating to the group and drawing from the group as an emergent, interactive context of being and staying in touch, physically or virtually.

> *"Is our body a thing or is it an idea? It is neither, being the measurement of the things. We will therefore have to recognize an ideality that is not alien to the flesh, that gives it its axes, its depth, its dimensions".*
>
> *Maurice Merleau-Ponty (1968: 152)*

References

Aberdour, M. 2007. "Achieving quality in open source software," IEEE Software (24:1), pp. 58-64.

Adams, R. M. 1976. "Motive Utilitarianism," The Journal of Philosophy (73:14), pp. 467-481.

Alaszewski, A., 2006. Using diaries for social research. London: Sage.

Alexy, O., and Leitner, M. 2011. "A fistful of dollars: Are financial rewards a suitable management practice for distributed models of innovation?" European Management Review. Forthcoming.

Alexy, O., P. Criscuolo, A. Salter. 2009. Does IP Strategy Have to Cripple Open Innovation? Mit Sloan Management Review 51(1) p.71-77.

Algesheimer, R., & Dholakia, P. M. 2006. Do customer communities pay off? Harvard Business Review, 84(11): 26.

Algesheimer, R., Dholakia, U. M., & Herrmann, A. 2005. The social influence of brand community: Evidence from European car clubs. Journal of Marketing, 69(3): 19-34.

Allen, Christopher. 2004. Tracing the evolution of social software. http://www.lifewithalacrity.com/2004/10/tracing_the_evo.html.

Allen, R.C. 1983. Collective invention. Journal of Economic Behavior & Organization 4(1) p.1.

Alpern, B., Augart, S., Blackburn, S.M., Butrico, M. , Cocchi, A., Cheng, P, Dolby, J., Fink, S., Grove, D., Hind, M., McKinley, K.S., Mergen, M., Moss, J.E.B., Ngo, T., Sarkar, V., M. Trapp. 2005. The Jikes Research Virtual Machine project: Building an open-source research community. IBM Systems Journal. 44(2). URL: http://www.research.ibm.com/journal/sj/442/alpern.pdf

Alvarez, S.A., Barney, J.B., 2005. How do entrepreneurs organize firms under conditions of uncertainty? Journal of Management 31 (5), 776–793.

Alvarez, S.A., Barney, J.B., 2007. Discovery and creation: alternative theories of entrepreneurial action. Strategic Entrepreneurship Journal 1 (1–2), 11–26.

Amin, A. and Roberts, J., 2008. Knowing in action: Beyond communities of practice'. Research Policy 37 (2): 353-369.

Anderson, C. J., Glassman, M., McAfee, R. B. and Pinelli , T., 2001. An investigation of factors affecting how engineers and scientists seek information. Journal of Engineering and Technology Management 18 (2): 131-155.

Andreoni, J. 1990. Impure Altruism and Donations to Public-Goods - a Theory of Warm-Glow Giving. Economic Journal 100(401) p.464-477.

Aoyama, Y., Izushi, H., 2003. Hardware gimmick or cultural innovation? Technological, cultural, and social foundations of the Japanese video game industry. Research Policy 32, 423–444.

Apte, U., C.S. Sankar, M. Thakur, J.E. Turner. 1990. Reusability-based strategy for development of information systems: Implementation experience of a bank. MIS Quarterly. December. 375-401.

Arakji Reina and Karl R. Lang," Digital Consumer Networks and Producer-Consumer Collaboration: Innovation and Product Development in the Video Game Industry," Journal of Management Information Systems, Vol. 24(4), 199-224, Fall 2007.

Ardichivili, A., Page, V. and Wentling, T. (2003). Motivation and barriers to participation in virtual knowledge-sharing communities of practice. Journal of Knowledge Management 7 (1): 64-77.

Argote, L. 1999. Organizational learning: Creating, retaining, and transferring knowledge. Kluwer, Norwell, MA.

Argyres, N. 1999. The impact of information technology on coordination: Evidence from the B-2 "Stealth" Bomber. Organization Science. 10(2) 162-180.

Aristotle. 2001. The Basic Works of Aristotle, (R. McKeon, ed.) Modern Library.

Armstrong, A., & Hagel, J. 1996. The real value of on-line communities. Harvard Business Review, 74(3): 134-&.

Armstrong, C., V. Sambamurthy. 1999. Information technology assimilation in firms: The influence of senior leadership and IT infrastructures. Information Systems Research 10(4) 304-327.

Arora, A., A. Gambardella. 2010. Chapter 15 - The Market for Technology. In Handbook of The Economics of Innovation, Vol. 1. North-Holland, 641-678.

Arora, A., Fosfuri, A., Gambardella, A., 2001. Markets for Technology: Economics of innovation and corporate strategy. MIT Press, Cambridge.

Arrow, K.J. 1984. Information and economic behavior, in Collected papers of Kenneth Arrow, Volume 4. Cambridge, MA: Belknap Press.

Baldwin, C., Hienerth, C., von Hippel, E., 2006. How user innovations become commercial products: a theoretical investigation and case study. Research Policy 35 (9), 1291–1313.

Baldwin, C., K. Clark. 2000. Design Rules, Volume 1, The Power of Modularity. MIT Press, Cambridge, MA.

Baldwin, C.Y. 2008. Where do transactions come from? Modularity, transactions, and the boundaries of firms. Industrial and Corporate Change. 17(1), 155-195.

Baldwin, C.Y., K.B. Clark. 2006. The architecture of participation: Does code architecture mitigate free riding in the open source development model? Management Science. 52(7) 1116-1127.

Banker, R., R. Kauffman. 2004. The evolution of research on information systems: A fiftieth-year survey of the literature in Management Science. Management Science 50(3) 281-298.

Banker, R.D., R,J. Kauffman, D. Zweig. 1993. Repository evaluation of software reuse. IEEE Transactionson Software Engineering. 19(4) 379-389.

Banker, R.D., R,J. Kauffman. 1991. Reuse and productivity in integrated computer-aided software engineering: An empirical study. MIS Quarterly. 15(3) 375-401.

Barachini, F. 2009. Cultural and social issues for knowledge sharing. Journal of Knowledge Management 13(1) 98-110.

Barad, K. 2003. Posthumanist performativity: toward an understanding of how matter comes to matter. Signs 28(3) 801–31.

Barad, K.M. 2007. Meeting the universe halfway: quantum physics and the entanglement of matter and meaning. Duke University Press.

Barley, SR. 1986. "Technology as an occasion for structuring - evidence from observations of CT scanners and the social-order of radiology departments." Administrative Science Quarterly 31 (1) (March): 78-108.

Barnes, B.C., T.B. Bollinger. 1991. Making reuse cost-effective. IEEE Software. 8(1) 13-24.

Barnes, T., T. Durek, J. Gaffney, A. Pyster. 1987. A framework and economic foundation for software reuse. Proc. Workshop Software Reuse and Maintainability.

Barwise, P, and S Meehan. 2010. "The One Thing You Must Get Right When Building a Brand." Harvard Business Review 88 (12) (December): 80-+.

Basili, V.R. 1990. Viewing maintenance as reuse-oriented software development. IEEE Software. 1 19-25.

Beadle, R., and Moore, G. 2006. "MacIntyre on Virtue and Organization," Organization Studies (27:3), pp. 323-340.

Bellemare, C., S. Kroeger, A. van Soest. 2008. Measuring inequity aversion in a heterogeneous population using experimental decisions and subjective probabilities. Econometrica 76(4), 815-839.

Bender, R. 1998, January. "The Aesthetics of Ethical Reflection and the Ethical Significance of Aesthetic Experience: A Critique of Alasdair MacIntyre and Martha Nussbaum," Erfurt Electronic Studies in English.

Benkler, Y. 2002. "Coase's Penguin, or Linux and the Nature of the Firm," Yale Law Journal (112:3), pp. 369-447.

Berends, H., Vanhaverbeke, W. and Kirschbaum, R., 2007. Knowledge management challenges in new business development: Case study observations. Journal of Engineering and Technology Management 24 (4): 314-328.

Bergquist M., J. Ljungberg. 2001. The power of gifts: organizing social relationships in open source communities. Information Systems Journal. 11 305-320.

Bernoff, J., & Li, C. 2008. Marketing - Harnessing the power of the oh-so-social web. Mit Sloan Management Review, 49(3): 36-+.

Bessen, J., E. Maskin. 2009. Sequential innovation, patents, and imitation. RAND Journal of Economics 40(4), 611 – 635.

Birkinshaw, J., C. Bouquet. 2011. The 5 Myths of Innovation. Mit Sloan Management Review 52(2) p.43-.

Bitzer, J., Schrettl, W., and Schröder, P. J. 2007. "Intrinsic Motivation in Open Source Software Development," Journal of Comparative Economics (35:1), pp. 160-169.

Blackledge, P. 2009. "Alasdair MacIntyre: Social Practices, Marxism and Ethical Anti-Capitalism," Political Studies (57:4), pp. 866-884.

Bodker, M., L. Nielsen, R. Orngreen. 2007. Enabling user Centered design processes in OS communities. Lecture Notes in Computer Science 4559 10-18.

Boldrin, M., D. Levine. 2001. The case against intellectual property. American Economic Review 92 209-212.

Boltanski, L., L. Thévenot. 2006. On justification: economies of worth. Princeton University Press.

Bonaccorsi A., C. Rossi. 2003. Why Open Source software can succeed. Research Policy. 32(7) 1243-1258.

Bonaccorsi, A., and Rossi, C. 2006. "Comparing Motivations of Individual Programmers and Firms to Take Part in the Open Source Movement: From Community to Business," Knowledge, Technology, and Policy (18:4), pp. 40-64.

Bonaccorsi, A., S. Giannangeli, C. Rossi. 2006. Entry strategies under competing standards: Hybrid business models in the OS software industry. Management Science 52(7) 1085-1098.

Borzillo, S. (2009). Top management sponsorship to guide communities of practice. Journal of Knowledge Management 13 (3): 60-72.

Boyd, D. M., & Ellison, N. B. 2007. Social network sites: Definition, history, and scholarship. Journal of Computer-Mediated Communication, 13(1): 210-230.

Braa, J., C. Hedberg. 2002. The struggle for district-based health information systems in South Africa. The Information Society 18(2) 113.

Brandts, J., G. Charness. 2000. Hot vs. cold: Sequential responses and preference stability in experimental games. Experimental Economics 2 227-238.

Brandts, J., G. Charness. 2003. Truth or Consequences: An Experiment. Management Science 49 116-130.

Brewer, K. B. 1997. "Management as a Practice: A Response to Alasdair MacIntyre," Journal of Business Ethics (16:8), pp. 825-833.

Britton, J.B.H., Tremblay, D.-G., Smith, R., 2009. Contrasts in clustering: the example of Canadian New Media. European Planning Studies 17 (2), 211–234.

Brockman, B.K., Morgan, R.M., 2003. The role of existing knowledge in new product innovativeness and performance. Decision Sciences 34 (2), 385–419.

Broothaerts W., Mitchell H.J., Weir B., Kaines S., Smith L.M.A., Yang W., Mayer J.E., Roa-Rodriguez C., Jefferson R.A. 2005. Gene transfer to plants by diverse species of bacteria. Nature 433 629-633.

Brown, J. S. and Duguid, P., 1991. Organizational learning and Communities-of-Practice: Toward a unified view of working, learning, and innovation. Organization Science 2 (1): 40-57.

Brown, J. S. and Duguid, P., 2001. Knowledge and organization: a social-practice perspective. Organisation Science 12 (2): 198-213.

Brown, S.L., Eisenhardt, K.M., 1997. The art of continuous change: linking complexity theory and time-paced evolution in relentlessly shifting organizations. Administrative Science Quarterly 42, 1–34.

Brusoni, S. 2005. The limits to specialization: Problem solving and coordination in 'modular networks'. Organization Studies. 26(12) 1885-1907.

Brusoni, S., A. Prencipe. 2006. Making Design Rules: A Multidomain Perspective. Organization Science. 17(2) 179-189.

Brusoni, S., Prencipe A., K. Pavitt. 2001. Knowledge specialization, organizational coupling, and the boundaries of the firm: Why do firms know more than they make? Administrative Science Quarterly. 46(4) 597-621.

Brynjolfsson E, and A Saunders. 2009. Wired for innovation: how information technology is reshaping the economy, MIT Press.

Brynjolfsson, E., L.M. Hitt. 1998. Beyond the productivity paradox. Commun. ACM 41(8) 49-55.

Brynjolfsson, E., L.M. Hitt. 2000. Beyond Computation: Information Technology, Organizational Transformation and Business Performance. The Journal of Economic Perspectives 14(4) p.23-48.

Butler, T., J. Feller, A. Pope, B. Emerson, C. Murphy. 2008. Designing a core IT artefact for Knowledge Management Systems using participatory action research in a government and a non-government organisation. The Journal of Strategic Information Systems 17(4) 249-267.

Cabrera, A. and Cabrera E.F., 2002. Knowledge-sharing dilemmas. Organization Studies 23 (5): 687-710.

Calantone, R.J., Yeniyurt, S., Townsend, J.D., Schmidt, J.B., 2010. The effects of competition in short product lifecycle markets: the case of motion pictures. Journal of Product Innovation Management 27, 349–361.

Calhoun, C. J. 1983. "The radicalism of tradition: Community strength or venerable disguise and borrowed language?," American Journal of Sociology (88:5), pp. 886–914.

Camerer, C. 2003. Behavioral game theory. Princeton: Princeton University Press.

Camerer, C., M. Knez, and R. Weber. 1997. Coordination in organizations: A game-theoretic perspective. Zur Shapira, (ed). Organizational Decision Making Cambridge Series on Judgment and Decision Making, Cambridge, U.K.

Capra, E, C Francalanci, F Merlo, and C Rossi-Lamastra. 2011. "Firms' involvement in Open Source projects: A trade-off between software structural quality and popularity." Journal of Systems and Software 84 (1) (January): 144-161.

Carbon, G., A. Lesniak, D. Stoddard. 2001. Open Source Enterprise Solutions: Developing an E Business Strategy. John Wiley & Sons, New York.

Carlile, P.R. 2004. Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. Organization science 15(5) p.555-568.

Casadesus Masanell, R., G. Llanes. 2011. Mixed Source. Management Science 57(7) p.1212-1230.

Casadesus-Masanell, R., P. Ghemawat. 2006. Dynamic mixed duopoly: A model motivated by Linux vs. Windows. Management Science. 52 1072-1084.

Casalo, L. V., Flavian, C., & Guinaliu, M. 2008. Promoting consumer's participation in virtual brand communities: A new paradigm in branding strategy. Journal of Marketing Communications, 14(1): 19-36.

Charness, G., M. Rabin. 2002. Understanding social preferences with simple tests. Quarterly Journal of Economics 117 817-869.

Chesbrough, H. 2003a. Towards a dynamics of modularity: A cyclical model of technical advance. In: Prencipe, A., Davies, A., M. Hobday (Eds.). The business of systems integration. Oxford University Press.

Chesbrough, H. 2003b. Open innovation. The new imperative for creating and profiting from technology. Harvard Business School Press.

Chesbrough, H.W., 2003. The era of open innovation. Sloan Management Review 44 (3), 35–41.

Chirico, F., Salvato, C., 2008. Knowledge integration and dynamic organizational adaptation in family firms. Family Business Review 21 (2), 169–181.

Chiu, C., M. Hsu, E. Wang. 2006. Understanding knowledge sharing in virtual communities: An integration of social capital and social cognitive theories. Decision Support Systems 42(3) 1872-1888.

Ciborra, C. 1998. Crisis and foundations: an inquiry into the nature and limits of models and methods in the information systems discipline. Journal of Strategic Information Systems 7(1) 5-16.

Ciborra, C. 2006. Imbrication of Representations: Risk and Digital Technologies. Journal of Management Studies 43(6) 1339-1356.

Ciborra, CU. 1996. "The platform organization: Recombining strategies, structures, and surprises." Organization Science 7 (2) (April): 103-118.

Clark, K.B. 1985. The interaction of design hierarchies and market concepts in technological evolution. Research Policy. 14 235-251.

Clark, K.B., T. Fujimoto. 1991. Product development performance: strategy, organization, and management in the world auto industry. Harvard Business Press.

Coase, R.H. 1937. The Nature of the Firm. Economica 4(16) p.386-405.

Cohen, B.P., 1980. Developing Sociological Knowledge: Theory and Method. Prentice Hall, Englewood Cliffs, NJ.

Cohendet, P., Simon, L., 2007. Playing across the playground: paradoxes of knowledge creation in the video-game firm. Journal of Organizational Behaviour 28 (5), 587–605.

Collier, Benjamin, Moira Burke, Niki Kittur, and Robert Kraut. 2010. "Promoting Good Management: Governance, Promotion, and Leadership in Open Collaboration Communities." ICIS 2010 Proceedings (January 1). http://aisel.aisnet.org/icis2010_submissions/220.

Colombo, M.G., M. Delmastro. 2002. How effective are technology incubators? Evidence from Italy. Research Policy 31(7) p.1103-1122.

Colquitt, JA, DE Conlon, MJ Wesson, COLH Porter, and KY Ng. 2001. "Justice at the millennium: A meta-analytic review of 25 years of organizational justice research." Journal of Applied Psychology 86 (3) (June): 425-445.

Comino, S., A. Rossi, F.M. Manenti. 2010. Public Intervention for Free/Open Source Software. SSRN eLibrary.

Cooke, M, and N Buckley. 2008. "Web 2.0, social networks and the future of market research." International Journal of Market Research 50 (2): 267-292.

Cooper, H. 1998. Synthesizing Research, (3rd ed.) Thousand Oaks: Sage Publication.

Cornman, J. W., Lehrer, K., and Pappas, G. S. 1982. Philosophical problems and arguments: an introduction, (3rd ed.) Indianapolis, Indiana: Hackett Publishing.

Cross, N. 2007. Forty years of design research. Design Studies 28(1) 1-4.

Cross, R., Laseter, T., Parker, A. and Velasquez, G., 2006.Using Social Network Analysis to Improve Communities of Practice. California Management Review 49 (1): 32-60.

Cross, R., P. Gray, S. Cunningham, M. Showers, R.J. Thomas. 2010. The Collaborative Organization: How to Make Employee Networks Really Work. MIT Sloan management review 52(1) p.83-.

Culnan, MJ, PJ McHugh, and JI Zubillaga. 2010. "How large U.S. companies can use Twitter and other social media to gain business value." MIS Quarterly Executive 9 (4) (December): 243-259.

Cusumano, M. 1991. Japan's software factories. Oxford University Press.

Cusumano, M.A., R.W. Selby. 1995. Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People. Free Press.

Cybulski, J.J., R.D. Neal, A. Kram, J.C. Allen. 1998. Reuse of early life-cycle artifacts: work products and methods, and tools. Annals of Software Engineering. 5 227-251.

da Cunha, João Vieira, and Wanda J. Orlikowski. 2008. "Performing catharsis: The use of online discussion forums in organizational change." Information and Organization 18 (2): 132-156.

Dahlander, L, and M Magnusson. 2008. "How do Firms Make Use of Open Source Communities?" Long Range Planning 41 (6) (December): 629-649.

Dahlander, L. 2007. Penguin in a new suit: a tale of how de novo entrants emerged to harness free and open source software communities. Industrial and Corporate Change. 16(5) 913–943.

Dahlander, L., M.W. Wallin. 2006. A man on the inside: Unlocking communities as complementary assets. Research Policy 35(8) 1243-1259.

Dasgupta, P. and P. A. David. 1994. Toward a New Economics of Science. Policy Research 23 487-521.

David, P. A., and Shapiro, J. S. 2008. "Community-based production of open-source software: What do we know about the developers who participate?," Information Economics and Policy (20:4), pp. 364-398.

David, P. A., Waterman, A., and Arora, S. 2003. FLOSS-US: The Free/Libre/Open Source Software Survey for 2003.

Dawson, D., and Bartholomew, C. 2003. "Virtues, Managers and Business People: Finding a Place for MacIntyre in a Business Context," Journal of Business Ethics (48:2), pp. 127-138.

De Hertogh, S., S. Viaene, G. Dedene. 2011. Governing Web 2.0. Communications of the ACM 54(3) p.124-130.

de Valck, K., Langerak, F., Verhoef, P. C., & Verlegh, P. W. J. 2007. Satisfaction with virtual communities of interest: Effect on members' visit frequency. British Journal of Management, 18(3): 241-256.

de Vries, R.E., van den Hooff B., de Ridder J.A. 2006. Explaining knowledge sharing - The role of team communication styles, job satisfaction, and performance beliefs. Communication Research. 33 115-135

Deci, E. L., and Ryan, R. M. 1985. Intrinsic Motivation and Self-Determination in Human Behavior, Plenum.

Deci, E. L., and Ryan, R. M. 1987. "The Support of Autonomy and the Control of Behavior," Journal of Personality and Social Psychology (53:6), pp. 1024-1037.

Dellarocas, C, and CA Wood. 2008. "The sound of silence in online feedback: Estimating trading risks in the presence of reporting bias." Management Science 54 (3) (March): 460-476.

Demil, B., X. Lecocq. 2006. Neither Market nor Hierarchy nor Network: The Emergence of Bazaar Governance. Organization Studies 27(10) 1447-1466.

Demsetz, H. 1967. Towards a theory of property rights. American Economic Review. 57 347-359.

Deng, S.Y., He, L., Xia, W.W. 2008. A Collaborative Filtering Algorithm Based on Rating Distribution. IEEE Proceedings. International Symposium on IT in Medicine and Education. 1-2: 1118-1122.

Depoorter, B., 2009. Technology and uncertainty: the shaping effect on copyright law. University of Pennsylvania Law Review 157, 1831–1868.

DeSanctis, G., M.S. Poole. 1994. Capturing the complexity in advanced technology use—adaptive structuration theory. Organization Science 5(2) 121-147.

Dholakia, U., Bagozzi, R., & Pearo, L. 2004. A social influence model of consumer participation in network- and small-group-based virtual communities. International Journal of Research in Marketing, 21(3): 241-263.

di Norcia, V. 2002. The knowledge economy and moral community. Journal of Business Ethics 38 167-177.

Dosi, G. 1988. Sources, procedures and microeconomic effects of innovation. Journal of Economic Literature. 26 1120-1171.

Dosi, G., Hobday, M., Marengo L., A. Prencipe. 2003. The economics of systems integration: Towards an evolutionary interpretation. In: Prencipe, A., Davies, A., M. Hobday

(Eds.). The business of systems integration. Oxford University Press.

Dufwenberg, M., G. Kirchsteiger. 2004. A theory of sequential reciprocity. Games and Economic Behavior 47 268-298.

Dunne, J. 1992. Back To the Rough Ground: "Phronesis" and "Techne" in Modern Philosophy and in Aristotle, (1st ed.) University of Notre Dame Press.

Economides, N. 1996. The economics of networks. International Journal of Industrial Organization 14(6) p.673-699.

Economides, Nicholas, E. Katsamakas. 2006. Two-sided competition of proprietary vs. open source technology platforms and the implications for the software industry. Management Science 52(7) p.1057-1071.

Economist 2011. Alternative uses: The play's the thing. The Economist. Available at: http://www.economist.com/node/21541159

Eisenhardt, K.M. 1989. Building theories form case study research. Academy of Management Review. 14(4) 532-550.

Eisenhardt, K.M., Graebner, M.E., 2007. Theory building from cases: opportunities and challenges. Academy of Management Journal 50 (1), 25–32.

Eisenmann, T., G. Parker, M. Van Alstyne. 2011. Platform Envelopment. Strategic Management Journal 32(12) p.1270-1285.

Elbanna, A. 2010. Rethinking IS project boundaries in practice: A multiple-projects perspective. Journal of Strategic Information Systems 19(1) 39-51.

Eliashberg, J., Elberse, A., Leenders, M.A.A.M., 2006. The motion picture industry: critical issues in practice, current research, and new research directions. Marketing Science 25 (6), 638–661.

Elster, J. 1986. An introduction to Karl Marx. Cambridge: Cambridge University Press.

Engelmann, D., M. Strobel. 2004. Inequality Aversion, Efficiency, and Maximin Preferences in Simple Distribution Experiments. American Economic Review 94 857-869.

Falk, A., U. Fischbacher. 2006. A theory of reciprocity. Games and Economic Behavior. 54 293-315.

Faraj, S., L. Sproull. 2000. Coordinating expertise in software development teams. Management Science. 46 1554-1568.

Faraj, S., M.M. Wasko. 2005. The web of knowledge: An investigation of knowledge exchange in networks of practice. Working paper: http://opensource.mit.edu/papers/Farajwasko.pdf

Faraj, S., S.L. Jarvenpaa, Ann Majchrzak. 2011. Knowledge Collaboration in Online Communities. Organization science 22(5) p.1224-1239.

Farjoun, M., 1994. Beyond industry boundaries: human expertise, diversification and resource-related industry groups. Organization Science 5 (2), 185–199.

Fauchart, E., E. von Hippel. 2006. Norms-based intellectual property systems: The case of French chefs. MIT Sloan School of Management Working Paper 4576-06.

Faulkner, J. Runde. 2010. The social, the material, and the ontology of non-material technological objects. Working paper. URL: http://www.lse.ac.uk/collections/informationSystems/newsAndEvents/ 2011events/Non-MaterialTechnologicalObjects.pdf

Faulkner, P., C. Lawson, J. Runde. 2010. Theorising technology. Cambridge Journal of Economics 34(1) p.1 -16.

Faulkner, P., Runde, J., 2009. On the identity of technological objects and user innovations in function. Academy of Management Review 34 (3), 442–462.

Favaro, J.M., K.R. Favaro, P.F. Favaro. 1998. Value-based software reuse investment. Annals of Software Engineering. 5 5-52.

Fehr, E., K. Schmidt. 1999. A theory of fairness, competition, and cooperation. Quarterly Journal of Economics 114(3) p.817-868.

Fehr, E., S Gächter. 2000. Fairness and retaliation: The economics of reciprocity. Journal of Economic Perspectives 14 159-181.

Feldman, M. S. 2000. Organizational routines as a source of continuous change. Organization Science 11(6) 611–629.

Feldman, M.S., Orlikowski, W.J. 2011. "Theorizing Practice and Practicing Theory," Organization Science, Articles in Advance, pp. 1–14.

Feldman, R.C. 2004. The Open Source Biotechnology Movement: Is it Patent Misuse?. Minnesota Journal of Law, Science & Technology 6 117-167.

Feller, J., and Fitzgerald, B. 2002. Understanding Open Source Software Development, Addison-Wesley.

Fershtman, C., and Gandal, N. 2007. "Open source software: Motivation and restrictive licensing," International Economics and Economic Policy (4:2), pp. 209-225.

Fischbacher, U. 2007. z-Tree, Toolbox for readymade economic experiments. Experimental Economics 10(2), 171-178.

Fisher, W.W. 2004. Promises to keep: technology, law, and the future of entertainment. Stanford University Press.

Fitzgerald, B. 2006. The transformation of OS software. MIS Quarterly 30(3) 587-598.

Fitzgerald, B., T. Kenny. 2004. Developing an information systems infrastructure with OS software. IEEE Software 21(1) 50.

Fleming, L, and DM Waguespack. 2007. "Brokerage, boundary spanning, and leadership in open innovation communities." Organization Science 18 (2) (April): 165-180.

Flick, U., 2002. An introduction to qualitative research (2nd ed.). Sage, London.

Flora, C.B. & Flora, J.L. (1993). Entrepreneurial social infrastructure - a necessary ingredient. Annals of the American Academy of Political and Social Science, 529(1), 48-58.

Flyvbjerg, B. 2001. Making Social Science Matter, (1st ed,)Cambridge University Press.

Fortes, M. 1969. Kinship and the social order: the legacy of Lewis Henry Morgan, Chicago: Aldine.

Foss, N.J. 1996. Knowledge-based approaches to the theory of the firm: Some critical comments. Organization Science 7 470-476.

Foss, N.J., V. Mahnke. 2003. Knowledge management: What can organizational economics contribute? M. Easterby-Smith and M. Lyles (eds). Handbook of Organizational Learning and Knowledge Management (pp. 78-104). Oxford: Blackwell.

Fowers, B. J. 2003. "Reason and Human Finitude: In Praise of Practical Wisdom," American Behavioral Scientist (47:4), pp. 415-426.

Frakes, W., S. Isoda. 1994. Success factors of systematic reuse. IEEE Software. 11(5) 15-19.

Frakes, W.B., C. Terry. 1996. Software reuse: Metrics and models. ACM Computing Surveys. 28(2) 415-35.

Francalanci, C., F. Merlo. 2008. Empirical Analysis of the Bug Fixing Process in Open Source Projects. In B. Russo, E. Damiani, S. Hissam, B. Lundell, G. Succi, eds. Open Source Development, Communities and Quality. Boston, MA: Springer US, 187-196.

Franke, N. and E. von Hippel. 2003. Satisfying heterogeneous user needs via innovation toolkits: The case of Apache security software. Research Policy. 32 1199-1216.

Franke, N., F. Piller. 2004. Value creation by toolkits for user innovation and design: The case of the watch market. The Journal of Product Innovation Management 21(6) p.401-415.

Franke, N., Shah, S., 2003. How communities support innovative activities: an exploration of assistance and sharing among end-users. Research Policy 32 (1), 157–178.

Franke, Nikolaus, Martin Schreier, and Ulrike Kaiser. 2010. "The 'I Designed It Myself' Effect in Mass Customization." Management Science 56 (1) (January 1): 125-140.

Fredberg, T. 2009. "Organising Customers: Learning from Big Brother." Long Range Planning 42 (3) (June): 320-340.

Frey, B. S., and Jegen, R. 2001. "Motivation crowding theory," Journal of Economic Surveys (15), pp. 589-610.

Friedman, B., and Kahn, P. H. 1994. "Educating computer scientists: linking the social and the technical," Communications of the ACM (37:1), pp. 64–70.

Fuchs, Christoph, and Martin Schreier. 2011. "Customer Empowerment in New Product Development." Journal of Product Innovation Management 28 (1) (January): 17-32.

Füller, J, R Faullant, and K Matzler. 2010. "Triggers for virtual customer integration in the development of medical equipment - From a manufacturer and a user's perspective." Industrial Marketing Management 39 (8) (November): 1376-1383.

Füller, J., Jawecki, G., Mühlbacher, H., 2007. Innovation creation by online basketball communities. Journal of Business Research 60 (1), 60–71.

Füller, J., K. Matzler, M. Hoppe. 2008. Brand community members as a source of innovation. Journal of Product Innovation Management 25(6) p.608-619.

Gächter, S., von Krogh, G., Haefliger, S. 2010. Initiating private-collective innovation: The fragility of knowledge sharing. Research Policy.

Gagné, M., and Deci, E. L. 2005. "Self-determination theory and work motivation," Journal of Organizational Behavior (26:4), pp. 331-362.

Gamma, E., R. Helm, R. Johnson, J. Vlissides. 1995. Design Patterns. Addison-Wesley.

Garud, R., A. Kumaraswamy. 1995. Technological and organizational designs for realizing economies of substitution. Strategic Management Journal. 16 93-109.

Garud, R., Jain, S., Kumaraswamy, A. 2002. Institutional entrepreneurship in the sponsorship of common technological standards: the case of Sun Microsystems and Java. Academy of Management Journal 45 196-214.

Garud, R., S. Jain, P. Tuertscher. 2008. Incomplete by design and designing for incompleteness. Organization Studies 29(3) p.351-371.

Gawer, A., M.A. Cusumano. 2008. How companies become platform leaders. Mit Sloan Management Review 49(2) p.28-.

Geiger, D. 2009. "Revisiting the Concept of Practice: Toward an Argumentative Understanding of Practicing," Management Learning (40:2), pp. 129-144.

Geroski, P.A., 2000. Models of technology diffusion. Research Policy 29 (4–5), 603–625.

Ghosh, R. A. 2005. "Understanding Free Software Developers: Findings from the FLOSS Study," In Perspectives on Free and Open Source Software, J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani (eds.), MIT Press, pp. 23-46.

Ghosh, R., Glott, R., Krieger, B., G. Robles. 2002. Free/Libre and Open Source software: Survey and Study. University of Maastricht. URL: http://www.infonomics.nl/FLOSS/report/

Ghoshal, S., P. Moran 1996. Bad for practice: A critique of the transaction cost theory. Academy of Management Review. 21(1) 13-47.

Gillies, A. C. 1992. Software quality: theory and management, London, New York: Chapman & Hall.

Glaser, B., Strauss, A.L., 1967. The Discovery of Grounded Theory. Aldine, Chicago, IL.

Gloor, P.A., S.M. Cooper. 2007. The new principles of a swarm business. Mit Sloan Management Review 48(3) p.81-.

Gramazio F., Kohler, M. 2008. Digital Materiality in Architecture. Lars Müller Publishers.

Grant, R. & Baden-Fuller, C, 2004. A knowledge accessing theory of strategic alliances. Journal of Management Studies, 41(1), pp.61-84.

Grant, R.M. 1996. Toward a Knowledge-Based Theory of the Firm. Strategic Management Journal 17 SI 109-122.

Gregor, S. 2006. The nature of theory in information systems. MIS Quarterly 30(3) 611-642.

Gregor, S., D. Jones. 2007. The anatomy of a design theory. Journal of the Association for Information Systems 8(5) 312-335.

Griss, M.L. 1993. Software reuse: From library to factory. IBM Systems Journal. 32(4) 548-566.

Guba, E. G. and Lincoln, Y.S., 1981. Effective evaluation: Improving the usefulness of evaluation results through responsive and naturalistic approaches. Jossey-Bass, San Francisco.

Habermas, J. 1988. On the Logic of the Social Sciences, (S. Weber Nicholsen and J. A. Stark, trans.) The MIT Press.

Hacklin, F., C. Marxt, F. Fahrni. 2006. Strategic venture partner selection for collaborative innovation in production systems: A decision support system-based approach. International Journal of Production Economics 104(1) p.100-112.

Haefliger, S, P Reichen, PM Jager, and G von Krogh. 2009. "Modding as Rating Behavior in Virtual Communities: The Case of Rooster Teeth Productions." Lecture Notes in Computer Science 5621: 197-206.

Haefliger, S., E. Monteiro, D. Foray, Georg von Krogh. Social Software and Strategy. Long Range Planning 44(5-6) p.297-316.

Haefliger, S., G. von Krogh, S. Spaeth. 2008. Code Reuse in OS Software. Management Science 54(1) 180-193.

Haefliger, S., Jäger, P., G. von Krogh, 2010. "Under the Radar: Industry Entry by User Entrepreneurs." Research Policy, 39, pp. 1198–1213.

Hagel, J., & Armstrong, A. 1997. Net Gain: Expanding Markets Through Virtual Communities: {Harvard Business School Press}.

Halliday, J., and Johnsson, M. 2010. "A MacIntyrian perspective on organizational learning," Management Learning (41:1), pp. 37-51.

Hancock, H., Ingram, J., 2007. Machinima for Dummies. Wiley, NY.

Hardin, G. 1968. Tragedy of Commons. Science 162(3859) p.1243-&.

Hargrave, T., and Van de Ven, A. H. 2006. "A Collective Action Model of Institutional Innovation," Academy of Management Review (31:4), pp. 864-888.

Harhoff, D., Henkel, J., von Hippel, E. 2003. Profiting from voluntary information spillovers: How users benefit by freely revealing their innovations. Research Policy. 32 1753-1769.

Hars, A., and Ou, S. 2002. "Working for free? Motivations for participating in open-source projects," International Journal of Electronic Commerce (6), pp. 25-39.

Haruvy, E., Prasad, A., and Sethi, S. 2003. "Harvesting Altruism in Open-Source Software Development," Journal of Optimization Theory and Applications (118:2), pp. 381-416.

Heider, F. 1958. The Psychology of Interpersonal Relations, Wiley.

Hemetsberger, A. 2004. "Fostering Cooperation on the Internet: Social Exchange Processes in Innovative Virtual Consumer Communities," extended abstract published in Advances in Consumer Research (Vol. 29), S. M. Broniarczyk and K. Nakamoto (eds.), Valdosta, GA: Association for Consumer Research, pp. 354-356.

Henkel, J. 2006. Selective revealing in open innovation processes: The case of embedded Linux. Research Policy. 35 953-969.

Hertel, G. 2002. "Management virtueller Teams auf der Basis sozialpsychologischer Modelle," In Sozialpsychologie Wirtschaftlicher Prozess, E. H. Witt (ed.), Lengerich, Germany: Pabst Publishers, pp. 172-202.

Hertel, Guido, Sven Niedner, and Stefanie Herrmann. 2003. "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel." Research Policy 32 (7) (July): 1159-1177.

Hess, C, and Ostrom, E. (eds.) 2007. Understanding knowledge as a commons, Cambridge, MA: MIT Press

Hevner, A.R. 2007. The Three Cycle View of Design Science Research. Scandinavian journal of information systems 19(2) 87.

Hevner, A.R., S.T. March, J. Park, S. Ram. 2004. Design science in Information Systems research. MIS Quarterly 28(1) 75-105.

Hickman, L. 1998. Reading Dewey: interpretations for a postmodern generation. Indiana University Press.

Hienerth, C., P. Keinz, C. Lettl. October. Exploring the Nature and Implementation Process of User-Centric Business Models. Long Range Planning 44(5-6) p.344-374.

Hildreth, P. and Kimble, C. (2000). Communities of Practice in the distributed international environment. Journal of Knowledge Management 4 (1): 27-38.

Himanen, P. 2001. The Hacker Ethic and the Spirit of the Information Age, London, UK: Secker & Warburg.

Ho, Joanna L. Y, Anne Wu, and Sean Xin Xu. 2011. "Corporate Governance and returns on information technology investment: evidence from an emerging market." Strategic Management Journal 32 (6) (June 1): 595-623.

Hodgkinson, Gerard P., Eugene Sadler-Smith, Lisa A. Burke, Guy Claxton, and Paul R. Sparrow. 2009. "Intuition in Organizations: Implications for Strategic Management." Long Range Planning 42 (3) 277-297.

Hoetker, G. 2006. Do modular products lead to modular organizations? Strategic Management Journal. 27 501-518.

Holmstrom B., P. Milgrom. 1994. The firm as an incentive system. American Economic Review 84 972-991.

Hope, J. 2008. Biobazaar: the open source revolution and biotechnology. Harvard University Press: Camridge MA.

Huber, G.P., 1982. Organizational information systems: Determinants of their performance and behavior. Management Science 28 (2): 138-155.

Huber, G.P., 1991. Organizational learning: The contributing processes and the literatures. Organization Science 2 (1): 88-115.

Hustad, E, and R Teigland. 2008. "Implementing social networking media and Web 2.0 in multinationals: Implications for knowledge management." Proceedings of the 9th European Conference on Knowledge Management: 323-331.

Hyvattinen, H. 2006. Interface standards and creating innovation markets - implications on SMEs in a technology programme. Technovation 26(2) 262-273.

Iivari, J. 2007. A paradigmatic analysis of information systems as a design science. Scandinavian journal of information systems 19(2) 39.

Indyk, D. & Rier, D.A. (1993). Grass-roots aids knowledge - implications for the boundaries of science and collective action. Knowledge-Creation Diffusion Utilization, 15(1), 3-43.

Isakowitz, T., R.J. Kauffman. 1996. Supporting search for reusable software objects. IEEE Transactions on Software Engineering. 22(6) 407-423.

Isoda, S. 1995. Experiences of a software reuse project. Journal of Systems Software. 30 171-186.

Izushi, H., Aoyama, Y., 2006. Industry evolution and cross-sectoral skill transfers: a comparative analysis of the video game industry in Japan, the United States, and the United Kingdom. Environment and Planning A 38, 1843–1861.

Jarvenpaa, S.L., Ann Majchrzak. 2010. Vigilant Interaction in Knowledge Collaboration: Challenges of Online User Participation Under Ambivalence. Information Systems Research 21(4) p.773-784.

Jefferson, R.A. 2006. Science as Social Enterprise: The CAMBIA BiOS Initiative. Innovations 1 13-44.

Jeppesen, L.B., L. Frederiksen. 2006. Why do Users Contribute to Firm-hosted User Communities? The case of computer-controlled music instruments. Organization Science 17 45-63.

Jeppesen, L.B., Molin, M.J., 2003. Consumers as co-developers: learning and innovation outside the firm. Technology Analysis & Strategic Management 15 (3), 363–383.

Jeppesen, Lars Bo, and Karim R Lakhani. 2010. "Marginality and Problem-Solving Effectiveness in Broadcast Search." Organization Science 21 (September): 1016–1033.

Jick, T.D. 1979. Mixing qualitative and quantitative methods: Triangulation in action. Administrative Science Quarterly. 24 602-611.

Joos, R. 1994. Software reuse at Motorola. IEEE Software. 11(9) 42-47.

Kan, S. 2002. Metrics and models in software quality engineering, Boston: Addison-Wesley.

Kaplan, Andreas M., and Michael Haenlein. 2010. "Users of the world, unite! The challenges and opportunities of Social Media." Business Horizons 53 (1): 59-68.

Kardaras, D., Karakostas, B., & Papathanassiou, E. 2003. The potential of virtual communities in the insurance industry in the UK and Greece. International Journal of Information Management, 23(1): 41-53.

Katz, M., C. Shapiro. 1994. Systems Competition and Network Effects. Journal of Economic Perspectives 8(2) p.93-115.

Ke, W., and Zhang, P. 2008. "Motivations for Participating in Open Source Software Communities: Roles of Psychological Needs and Altruism," In 12th Pacific Asia Conference on Information Systems (PACIS 2008), pp. 136-148.

Kellogg, K. C., Orlikowski, W. J., and Yates, J. 2006. "Life in the Trading Zone: Structuring Coordination Across Boundaries in Postbureaucratic Organizations," Organization Science (17:1), pp. 22-44.

Kerner, S.M. 2010. You Can't Control Linux. CIO Update. http://www.cioupdate.com/features/article.php/3876581/You-Cant-Control-Linux.htm

Ketchen, D.J., R.D. Ireland, C.C. Snow. 2008. Strategic Entrepreneurship, Collaborative Innovation, and Wealth Creation. Strategic Entrepreneurship Journal 1(3-4) p.371-385.

Kim, W.C., R. Mauborgne. 1998. Procedural justice, strategic decision making, and the knowledge economy. Strategic Management Journal 19 323-338.

Kim, Y. A., Le, M. T., Lauw, H. W., Lim, E. P., Liu, H. F., Srivastava, J., & Ieee. 2008. Building a web of trust without explicit trust ratings. Paper presented at the 24th IEEE International Conference on Data Engineering, Cancun, MEXICO.

Kim, Y., E.A. Stohr. 1998. Software reuse: survey and research directions. Journal of Management Information Systems. 14(4) 113-147.

King, L. 2010. London Stock Exchange Linux record breaking system faces new challengers. Computerworld UK. Available at: http://www.computerworlduk.com/in-depth/open-source/3246835/london-stock-exchange-linux-record-breaking-system-faces-new-challengers/ [Accessed March 30, 2011].

Kirsch, L. J. 1996. "The Management of Complex Tasks in Organizations: Controlling the Systems Development Process," Organization Science (7:1), pp. 1-21.

Klandermans, B. 1997. The Social Psychology of Protest, Oxford, UK: Basil Blackwell.

Kling, R., and W. Scacchi. 1982. "The web of computing: Computer technology as social organization." Advances in computers 21: 1.

Knight, J.C., M.F. Dunn. 1998. Software quality through domain-driven certification. Annals of Software Engineering. 5 293-315.

Knight, K. 1998. The MacIntyre Reader, University of Notre Dame Press.

Kodama, M. 2006. Knowledge-based view of corporate strategy. Technovation 26(12) 1390-1406.

Kodama, M., 2007. Innovation and knowledge creation through leadership-based strategic community: Case study on high-tech company in Japan. Technovation 27 (3): 115-132.

Kogut, B., A. Metiu. 2001. Open-source software development and distributed innovation. Oxford Review of Economic Policy 17 248-264.

Kogut, B., U. Zander. 1996. What firms do? Coordination, identity, and learning. Organization Science 7 502-518.

Kogut, B., Zander, U., 1992. Knowledge of the firm, combinative capabilities, and the replication of technology. Organization Science 3 (3), 383–397.

Kohler, T., J. Fueller, K. Matzler, and D. Stieger. 2011. "Co-Creation in Virtual Worlds: The Design of the User Experience." MIS Quarterly. Forthcoming.

Kozinets, R. V. 1999. E-tribalized marketing?: the strategic implications of virtual communities of consumption. European Management Journal, 17(3): 252-264.

Kozinets, R., and J. Handelman. 2004. "Adversaries of consumption: Consumer movements, activism, and ideology." The Journal of consumer research 31 (3): 691-704.

Kozinets, R.V. 2002. The field behind the screen: Using netnography for marketing research in online communities. Journal of Marketing Research. 39(February) 61-72.

Krebs, D. L. 1970. "Altruism - Examination of Concept and a Review of Literature," Psychological Bulletin (73:4), pp. 258-302.

Krishnamurthy, S. 2002. Cave or community? An empirical examination of 100 mature open source projects. First Monday. 7(6).

Kroah-Hartman, G., Corbet, J., McPherson, A. 2009. Linux Kernel Development: How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It: An August 2009 Update URL: http://www.linuxfoundation.org/publications/whowriteslinux.pdf

Krüger, H., 1983. Gruppendiskussion. Überlegungen zur Rekonstruktion sozialer Wirklichkeit aus der Sicht der Betroffenen. Soziale Welt 34 (1): 90-109.

Kuk, G. 2006. Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. Management Science. 52 1031-1042.

Lakhani, K. R., and Wolf, R. G. 2005. "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects," In Perspectives on Free and Open Source Software, J. Feller, B. Fitzgerald, S. Hissam, and K. R. Lakhani (eds.), MIT Press, pp. 3-22.

Lakhani, K.R., E. von Hippel. 2003. How open source software works: „Free" user-to-user assistance. Research Policy. 32(6) 923-43.

Lamastra, C R. 2009. "Software innovativeness. A comparison between proprietary and Free/Open Source solutions offered by Italian SMEs." R & D management 39 (2): 153.

Lang, K.R., Di. Shang, and Roumen Vragov, "Designing Markets for Co-Production of Digital Culture Products", Decision Support Systems, Vol. 48(1), 33-45, December 2009.

Langlois, R.N. 1999. "Scale, scope, and the reuse of knowledge" in Dow, S.C., P.E. Earl (Eds). Economic Organization and Economic Knowledge. Edward Elgar, Cheltenham. 239-254.

Latour, B. 1996. Aramis, or the Love of Technology, Harvard University Press.

Latour, B. 1999. Pandora's Hope: Essays on the Reality of Science Studies, (1st ed.) Harvard University Press.

Lattemann, C., and Stieglitz, S. 2005. "Framework for Governance in Open Source Communities," In Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)IEEE Computer Society, pp. 192b.

Laursen, K, and A Salter. 2006. "Open for innovation: The role of openness in explaining innovation performance among UK manufacturing firms." Strategic Management Journal 27 (2) (February): 131-150.

Lave, J., 1988. Cognition in practice: Mind, mathematics, and culture in everyday life. Cambridge University Press, New York.

Lave, J., and Wenger, E. 1991. Situated Learning: Legitimate Peripheral Participation, Cambridge University Press.

Lawrence, T.B., Hardy, C., Phillips, N., 2002. Institutional effects of interorganizational collaboration: the emergence of proto-institutions. Academy of Management Journal 45 (1), 281–290.

Lee, E., 2008. Warming up to user-generated content. University of Illinois Law Review 5, 1458–1548.

Lee, G., R. Cole. 2003. From a firm-based to a community-based model of knowledge creation: The case of the Linux kernel development. Organization Science 14(6) 633-649.

Leidner, D., D. Preston, and D. Chen. 2010. "An examination of the antecedents and consequences of organizational ICT innovation in hospitals." Journal of strategic information systems 19 (3): 154-170.

Leonard-Barton, D. 1995. Wellsprings of knowledge. Harvard Business School Press.

Leonardi, P. 2007. "Activating the informational capabilities of information technology for organizational change." Organization science 18 (5): 813-831.

Leonardi, P. 2008. "Indeterminacy and the discourse of inevitability in international technology management." The Academy of Management review 33 (4): 975-984.

Leonardi, P.M. 2011. When Flexible Routines Meet Flexible Technologies: Affordance, Constraint, and the Imbrication of Human and Material Agencies. MIS Quarterly 35(1) p.147-167.

Leonardi, PM, and SR Barley. 2010. "What's Under Construction Here? Social Action, Materiality, and Power in Constructivist Studies of Technology and Organizing." Academy of Management Annals 4: 1-51.

Lerner, J. and J. Tirole. 2005. The Scope of Open Source Licensing. Journal of Law, Economics, and Organization. 21(1):20-56.

Lerner, J., J. Tirole. 2002. Some simple economics of open source. Journal of Industrial Economics. 52 197-234.

Lesser, E.L. and Storck, J., 2001. Communities of practice and organisational performance. IBM Systems Journal 40 (4): 831- 841.

Lessig, L. 2004. Free Culture. The Nature and Future of Creativity. The Penguin Press.

Levy, S. 1984. Hackers: Heroes of the Computer Revolution, Anchor Press/Doubleday.

Leyshon, A., 2001. Time-space (and digital) compression: software formats, musical networks, and the reorganization of the music industry. Environment and Planning A 33, 49–77.

Lieberman, M.B., 1987. The learning curve, diffusion, and competitive strategy. Strategic Management Journal 8 (5), 441–452.

Liebeskind, J.P., 1996. Knowledge, strategy, and the theory of the firm. Strategic Management Journal 17 (Winter Special Issue), 93–107.

Lim, W.C. 1994. Effects of reuse on quality, productivity, and economics. IEEE Software. 11(9) 23-30.

Lincoln, Y.S., E.G. Guba. 1985. Naturalistic inquiry. Sage Publications.

Lindenberg, S. M. 2001. "Intrinsic Motivation in a New Light," Kyklos (54:2/3), pp. 317-342.

Lingo, E.L., S. O'Mahony. 2010. Nexus Work: Brokerage on Creative Projects. Administrative Science Quarterly 55(1) p.47-81.

Lowood, 2007. High-performance play: the making of Machinima. In: Clarke, A., Mitchell, G. (Eds.), Videogames and Art. Intellect Books, Bristol.

Luhmann, N., 2000. Organisation und Entscheidung. Westdeutscher Verlag, Opladen.

Luthiger, B., and Jungwirth, C. 2007. "Pervasive fun," First Monday (12:1).

Luthje, C., Herstatt, C., von Hippel, E., 2005. User-innovators and "local" information: the case of mountain biking. Research Policy 34, 951–965.

Lynex, A., P. Layzell. 1998. Organisational Considerations For Software Reuse. Annals of Software Engineering 5, 105-124.

Lyytinen, K., J.L. King. 2006. Standard making: A critical research frontier for information systems research. MIS Quarterly 30 p.405-411.

MacCormack, A., J. Rusnak, C.Y. Baldwin. 2006. Exploring the structure of complex software designs: An empirical study of open source and proprietary code. Management Science. 52(7) 1015-1030.

MacCormack, A., M. Iansiti. 2009. Intellectual Property, Architecture, and the Management of Technological Transitions: Evidence from Microsoft Corporation. Journal of Product Innovation Management 26(3) 248-263.

MacIntyre, A. 1994. "Interview with Professor Alasdair MacIntyre," Kinesis (20), pp. 34-47.

MacIntyre, A. 1998a. "Politics, Philosophy and the Common Good," K. Knight (ed.), University of Notre Dame Press, pp. 235–252.

MacIntyre, A. 1998b. "The Claims of After Virtue," In The Macintyre Reader, K. Knight (ed.), University of Notre Dame Press, pp. 69-72.

MacIntyre, A. C. 1981. After virtue: A study in moral theory, (1st ed,)University of Notre Dame Press.

MacIntyre, A. C. 1999. Dependent Rational Animals: Why Human Beings Need the Virtues, Open Court Publishing Company.

Mackenzie, A. 2005. "The performativity of code: software and cultures of circulation," Theory, Culture & Society (22:1), pp. 71–92.

Maillart, T., D. Sornette, S. Spaeth, G. von Krogh. 2008. Empirical Tests of Zipf's Law Mechanism in Open Source Linux Distribution. Physical Review Letters 101(21).

Majchrak, A., L.P. Cooper, O.P. Neece. 2004. Knowledge reuse for innovation. Management Science. 50(2) 174-188.

Majchrzak, A., Wagner, C., and Yates, D. N. 2006. Corporate Wiki Users: Results of a Survey, in Proceedings of the 2006 International Symposium on Wikis, New York: ACM Press, pp. 99-104. URL: http://www.wikisym.org/ws2006/proceedings/p99.pdf

Maldonado, E. 2010. "The Process of Introducing FLOSS in the Public Administration: The Case of Venezuela." Journal of the Association for Information Systems 11(11): 756-783.

Malhotra, A., A. Majchrzak, R. Carman, V. Lott. 2001. Radical innovation without collocation: A case study at Boeing-Rocketdyne. Management Information Systems Quarterly 25(2) p.229-249.

March, J. 1991. Exploration and Exploitation in Organizational Learning, Organization Science, 2 71-87.

March, J.G., Sproull, L.S., Tamuz, M., 1991. Learning from samples of one of fewer. Organization Science 2 (1), 1–13.

March, S., G. Smith. 1995. Design and natural-science research on information technology. Decision Support Systems 15(4) 251-266.

March, S.T., V.C. Storey. 2008. Design science in the information systems discipline: An introduction to the special issue on design science research. MIS Quarterly 32(4) 725-730.

Marino, P., 2004. 3D Game-Based Filmmaking: The Art of Machinima. Paraglyph Press, Scottsdale, AZ.

Markus, M. 2007. "The governance of free/open source software projects: monolithic, multidimensional, or configurational?" Journal of Management and Governance 11 (2) (May 1): 151-163.

Markus, M. Lynne, and Daniel Robey. 1988. "Information Technology and Organizational Change: Causal Structure in Theory and Research." Management Science 34 (5) (May 1): 583-598.

Markus, M. Lynne. 1983. "Power, politics, and MIS implementation." Communications of the ACM 26 (June): 430–444.

Markus, M.L. 2001. Towards a theory of knowledge reuse: Types of knowledge reuse situations and factors in reuse success. Journal of Management Information Systems. 18(1) 57-93.

Markus, M.L., A. Majchrzak, L. Gasser. 2002. A Design Theory for Systems That Support Emergent Knowledge Processes. MIS Quarterly 26(3) 179-212.

Markus, M.L., M.S. Silver. 2008. A Foundation for the Study of IT Effects: A New Look at DeSanctis and Poole's Concepts of Structural Features and Spirit. Journal of the Association for Information Systems 9(10) 609-632.

Martin, M. W. 2000. Meaningful Work: Rethinking Professional Ethics, Oxford University Press.

Martin, M. W. 2002. "Personal meaning and ethics in engineering," Science and Engineering Ethics (8:4), pp. 545-560.

Martins, Luis L., Lucy L. Gilson, and M. Travis Maynard. 2004. "Virtual Teams: What Do We Know and Where Do We Go From Here?" Journal of Management 30 (6) (November): 805-835.

Massey, A., M. Montoya-Weiss, T. O'Driscoll. 2002. Knowledge management in pursuit of performance: Insights from Nortel Networks. MIS Quarterly 26(3) 269-289.

Mata, F., W. Fuerst, J. Barney. 1995. Information technology and sustained competitive advantage: A resource-based analysis. MIS Quarterly 19(4) 487-505.

Mauss, M. 1959. The gift. The form and the reason for exchange in archaic societies, Routledge.

Maxwell, J.C., 2005. Qualitative research design: An interactive approach. Sage Publications, Thousand Oaks.

McCarthy, J. & Zald, M.N. (1977). Resource mobilization and social-movements - partial theory. American Journal of Sociology, 82(6), 1212-1241.

McClure, C. 2001. Software reuse: A standards-based guide. IEEE Computer Society.

McGrath, RG. 2010. "Business Models: A Discovery Driven Approach." Long Range Planning 43 (2-3) (June): 247-261.

Meloso, D., J. Copic, P. Bossaerts. 2008. Promoting intellectual discovery: patents vs. markets. Science 323, 1335-1339.

Melucci, A. 1999. Challenging Codes: Collective Action in the Information Age. Cambridge University Press, Cambridge, U.K.

Merleau-Ponty, M. 1968. The visible and the invisible. Northwestern University Press.

Mezias, J.M., Mezias, S,J., 2000. Resource Partitioning, the Foundation of Specialist Firms and Innovation: the American Feature Film Industry, 1912–1929. Organization Science 11 (3), 306–322.

Michael, S., Storey, D., Thomas, H., 2002. Discovery and coordination in strategic management and entrepreneurship. In: Hitt, M., Ireland, R.D., Camp, S.M., Sexton, D.L. (Eds.), Strategic Entrepreneurship. Blackwell, Oxford, pp. 45–65.

Michailova, S. and Husted, K., 2003. Knowledge-sharing hostility in Russian firms. California Management Review 45 (3): 59-77.

Miles, M.B. and Huberman A.M., 1985. Qualitative data analysis. A source book of new methods. Sage Publications, Thousand Oaks.

Mill, J. S., Bentham, J., and Ryan, A. 1987. Utilitarianism and other essays, Penguin Classics.

Miller, R., Hobday, M., Leroux-Demers, T., X. Olleros. 1995. Innovation in complex systems industries: The case of flight simulation. Industrial and Corporate Change. 4(2) 363-400.

Mingers, J., G. Walsham. 2010. "Toward ethical information systems: The contribution of discourse ethics." MIS Quarterly 34(4): 833-854.

Mintzberg, H. 1978. "Patterns in Strategy Formation." Management Science 24 (9): 934-948.

Monteiro, Marko. 2010. Reconfiguring Evidence: Interacting with Digital Objects in Scientific Practice. Computer Supported Cooperative Work 19 335-354.

Monteverde, K. 1995. Technical dialog as an incentive for vertical integration in the semiconductor industry. Management Science. 41(10) 1624-1638.

Montmarquette, C., J.L. Rullière, M.C. Villeval, R. Zeiliger. 2004. Redesigning Teams and Incentives in a Merger: An Experiment with Managers and Students. Management Science. 50 1379-1389.

Moody, G. 2001. Rebel Code: Linux and the Open Source Revolution, London,UK: Penguin.

Moon, J.Y., L. Sproull. 2000. Essence of distributed work: The case of the Linux kernel. First Monday. 5(11).

Moore, G. 2008. "Re-imagining the morality of management: a modern virtue ethics approach.," Business Ethics Quarterly (18:4), pp. 483–511.

Morgeson, F. P., and Humphrey, S. E. 2006. "The Work Design Questionnaire (WDQ): Developing and validating a comprehensive measure for assessing job design and the nature of work," Journal of Applied Psychology (91:6), pp. 1321–1339.

Morris, A. D., and Mueller, C. M. 1992. Frontiers in Social Movement Theory, New Haven, CT: Yale University Press.

Morris, D., Kelland, M., Lloyd, D., 2005. Machinima. Course Technology PTR.

Morrison, P.D., Roberts, J.H., E. von Hippel. 2000. Determinants of user innovation and innovation sharing in a local market. Management Science. 46(12) 1513-1527.

Mowery, D.C. 1983. The relationship between intrafirm and contractual forms of industrial research in American manufacturing, 1900-1940. Explorations in Economic History 20(4) p.351.

Mulgan, G., Salem, O., T. Steinberg. 2005. Wide Open: Open source methods and their future potential. Demos Open Access. URL: http://www.demos.co.uk/ WideOpen_pdf_media_public.aspx

Muller-Seitz, G., G. Reger. 2010. Networking beyond the software code? an explorative examination of the development of an open source car project. Technovation 30(11-12) 627-634.

Muniz, AM, and TC O'Guinn. 2001. "Brand community." Journal of Consumer Research 27 (4) (March): 412-432.

Myers, D.J. (1994). Communication technology and social-movements - contributions of computer-networks to activism. Social Science Computer Review, 12(2), 250-260.

Nambisan, S. 2002. Designing virtual customer environments for new product development: Toward a theory. Academy of Management Review, 27(3): 392-413.

Nicolini, D. 2011. Practice as the Site of Knowing: Insights from the Field of Telemedicine. Organization Science 22(3) p.602-620.

Noel, J. 1999. "Phronesis and Phantasia: Teaching with Wisdom and Imagination," Journal of the Philosophy of Education (33:2), pp. 277-286.

Nonaka I, R. Toyama, T. Hirata. 2008. Managing Flow. A Process Theory of the Knowledge-Based Firm. Palgrave Macmillan.

Nonaka, I., 1994. A dynamic theory of organizational knowledge creation. Organization Science 5 (1): 14-37.

Nonaka, I., G. von Krogh. 2009. Tacit knowledge and knowledge conversion: Controversy and advancement in

organizational knowledge creation theory. Organization Science 20(3) 635-652.

Nonnecke, B. and Preece, J. 2000. Lurker demographics: Counting the silent, The Hague, ACM. Proceedings of CHI 2000.

Nussbaum, M. 1985. "Finely Aware and Richly Responsible: Moral Attention and the Moral Task of Literature," The Journal of Philosophy, pp. 516–529.

O'Mahony, S. 2003. Guarding the commons: How community-managed software projects protect their work. Research Policy. 32(7) 1179-1198.

O'Mahony, S, and BA Bechky. 2008. "Boundary Organizations: Enabling Collaboration among Unexpected Allies." Administrative Science Quarterly 53 (3) (September): 422-459.

O'Mahony, S., Ferraro F. 2007. The emergence of governance in an Open Source community. Academy of Management Journal. 50(5) 1079-1106.

O'Reilly, Tim. 2005. "What is Web 2.0?" URL: http://oreilly.com/web2/archive/what-is-web-20.html

O'Rourke, M.A., 2000. Toward a doctrine of fair use in patent law. Columbia Law Review, 100 (5): 1177–1250.

Ocasio, W. 1997. Towards an Attention-Based View of the Firm, Strategic Management Journal, 18(S1), 187-206.

Ogawa, S., Piller, F.T., 2006. Reducing the risks of new product development. MIT Sloan Management Review 47 (2), 65–71.

Okoli, C., and Oh, W. 2007. "Investigating recognition-based performance in an open content community: A social capital perspective," Information & Management (44:3), pp. 240-252.

Oliver, Pamela E. 1993. "Formal Models of Collective Action." Annual Review of Sociology 19: 271-300.

Olson, M. 1965. The Logic of Collective Action. Cambridge, MA: Harvard University Press.

Oreg, S., and Nov, O. 2008. "Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values," Computers in Human Behavior (24:5), pp. 2055-2073.

Orlikowski, W. 2010. The sociomateriality of organisational life: considering technology in management research. Cambridge Journal of Economics 34(1) p.125-141.

Orlikowski, W. J. 2000. "Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations," Organization Science (11:4), pp. 404-428.

Orlikowski, W., C.S. Iacono. 2001. Research commentary: Desperately seeking the „IT" in IT research – A call to theorizing the IT artifact. Information Systems Research. 12(2) 121-134.

Orlikowski, W., S. Scott. 2008. Sociomateriality: Challenging the Separation of Technology, Work and Organization. The Academy of Management Annals 2 p.433-474.

Orlikowski, W.J. 1992. Learning from notes: Organizational issues in groupware implementation. MIT Sloan Working Paper #3428-92. URL: http://ccs.mit.edu/papers/CCSWP134.html

Orlikowski, W.J. 1992. The duality of technology - rethinking the concept of technology in organizations. Organization Science 3(3) 398-427.

Orton, J.D., K.E. Weick. 1990. Loosely coupled systems: A reconceptualization. Academy of Management Review. 15(2) 203-223.

Osterloh, M, B.S. Frey. 2000. Motivation, Knowledge Transfer, and Organizational Forms, Organization Science 11 538-550.

Osterloh, M., S. Rota. 2007. Open source software development—Just another case of collective invention? Research Policy 36(2) p.157-171.

Ostrom, E. 1999. Coping with tragedies of the commons. Annual Review of Political Science 2(1) p.493-535.

Ostrom, Elinor. 1998. "A Behavioral Approach to the Rational Choice Theory of Collective Action: Presidential Address, American Political Science Association, 1997." The American Political Science Review 92 (1) (March 1): 1-22.

Ouchi, W. 1980. "Markets, Bureaucracies, and Clans." Administrative Science Quarterly 25(1): 129-141.

Ouchi, W. G. 1978. "The Transmission of Control through Organizational Hierarchy," The Academy of Management Journal (21:2), pp. 173-192.

Ouchi, W. G. 1979. "A Conceptual Framework for the Design of Organizational Control Mechanisms," Management Science (25:9), pp. 833-848.

Papies, D., Clemet, M., 2008. Adoption of new movie distribution services on the Internet. Journal of Media Economics 21, 131–157.

Pascu, C., Osimo, D. Ulbrich, M., Turlea, G. and J.C. Burgelman. 2007. The potential disruptive impact of Internet 2 based technologies. First Monday, 12(3).

Patton, M.Q., 2002. Qualitative research and evaluation methods. Sage Publications, Thousand Oaks.

Pauleen, DJ, and P Yoong. 2001. "Relationship building and the use of ICT in boundary-crossing virtual teams: a facilitator's perspective." Journal of Information Technology 16 (4) (December): 205-220.

Peretti, N., Negro, G., 2007. Mixing genres and matching people: a study in innovation and team composition in Hollywood. Journal of Organizational Behavior 28 (5), 563–586.

Perrow, C. 1967. "A framework for the comparative analysis of organizations." American Sociological Review 32 (2): 194.

Pervez, N.G., Tarnovskaya, V., Elg, U., 2008. Market driving multinationals and their global sourcing network. International Marketing Review 25 (5), 504–519.

Peters, L. 2003. Managing software professionals. In Engineering Management Conference, 2003. IEMC '03. Managing Technologically Driven Organizations: The Human Side of Innovation and Change. 61-66.

Pettigrew, A. 1988. Longitudinal field research on change. Theory and practice. Paper presented at the National Science Foundation Conference on Longitudinal Research Methods in Organizations, Austin, Texas.

Pisano, G.P. 1991. The Governance of Innovation - Vertical Integration and Collaborative Arrangements in the Biotechnology Industry. Research Policy 20(3) p.237-249.

Piven, F.F. & Cloward, R.A. (1992). Normalizing collective protest. In A. Morris & C. McClurg Mueller, (Eds.). Frontiers in social movement theory, (pp. 301-325). New Haven, CT: Yale University Press.

Poetz, MK, and R Prugl. 2010. "Crossing Domain-Specific Boundaries in Search of Innovation: Exploring the Potential of Pyramiding." Journal of Product Innovation Management 27 (6) (November): 897-914.

Porter, C. 2004. A Typology of Virtual Communities: A Multi-Disciplinary Foundation for Future Research. Journal of Computer-Mediated Communication, 10(1): 00-00.

Porter, C. E., & Donthu, N. 2008. Cultivating trust and harvesting value in virtual communities. Management Science, 54(1): 113-128.

Porter, M.E., 1980. Competitive Strategy. Techniques for Analyzing Industries and Competitors. Free Press, New York.

Poulin, J.S. 1995. Populating software repositories: Incentives and domain-specific software. Journal of Systems Software. 30 187-199.

Powell, TC, and A DentMicallef. 1997. "Information technology as competitive advantage: The role of human, business, and technology resources." Strategic Management Journal 18 (5) (May): 375-405.

Powell, W.W., K.W. Koput, L. Smithdoerr. 1996. Interorganizational collaboration and the locus of innovation: Networks of learning in biotechnology. Administrative Science Quarterly 41(1) p.116-145.

Prandelli, E., G. Verona, D. Raccagni. 2006. Diffusion of Web-based product innovation. California Management Review 48(4) p.109-.

Prencipe, A. 2000. Breadth and depth of technological capabilities in CoPS: the case of the aircraft engine control system. Research Policy. 29(7-8) 895-911.

Prencipe, A. 2003. Corporate strategy and systems integration capabilities: Managing networks in complex systems industries. In: Prencipe, A., Davies, A., M. Hobday (Eds.). The business of systems integration. Oxford University Press.

Prieto-Diaz, R. 1993. Status report: Software reusability. IEEE Software. 10(3) 61-66.

Rabin, M. 1993. Incorporating fairness into game theory and economics. American Economic Review 83 1281-1302.

Rämö, H. 2004. "Spatio-temporal notions and organized environmental issues: an axiology of action," Organization (11), pp. 849–872.

Rangachari, P. 2009. Knowledge sharing networks in professional complex systems. Journal of Knowledge Management 13(3) 132-145.

Ransbotham, S., G.C. Kane. 2011. Membership Turnover and Collaboration Success in Online Communities: Explaining Rises and Falls from Grace in Wikipedia. Management Information Systems Quarterly 35(3) p.613-627.

Ravichandran, T. 1999. Software reusability as synchronous innovation: A test of four theoretical models. European Journal of Information Systems. 8 183-199.

Ravichandran, T., Rothenberger, M.A. 2003. Software reuse strategies and component markets. Communications of the ACM. 46(8) 109-114.

Raymond, E. 1999. "The cathedral and the bazaar." Knowledge, Technology & Policy 12 (3): 23–49.

Raymond, E.S. 1998. Homesteading the Noosphere. First Monday. 3(10).

Read, S., Song, M., Smit, W., 2009. A meta-analytic review of effectuation and venture performance. Journal of Business Venturing 24 (6), 573–587.

Rehn, A. 2001. Electronic Potlatch - a study on new technologies and primitive economic behaviors. KTH Indek.

Ren, YQ, R Kraut, and S Kiesler. 2007. "Applying common identity and bond theory to design of online communities." Organization Studies 28 (3) (March): 377-408.

Rheingold, H. 1993. The Virtual Community: Homesteading on the Electronic Frontier: {Addison-Wesley Pub. Co., Reading, Mass.}.

Riehle, D. 2007. "The Economic Motivation of Open Source Software: Stakeholder Perspectives," Computer (40:4), pp. 25-32.

Rindova, V., Barry, D., Ketchen, D.J., 2009. Introduction to Special Topic Forum: entrepreneuring as emancipation. Academy of Management Review 34 (3), 477–491.

Roberts, J., Hann, I.-H. and S. Slaughter. 2006. Understanding the Motivations, Participation and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. Management Science. 52 984-999.

Robey, D., and Markus, L. M. 1984. "Rituals in Information System Design," MIS Quarterly (8:1), pp. 5-15.

Rogers, E.M., 1962. Diffusion of Innovations. Free Press, Glencoe.

Rolandsson, B, M Bergquist, and J Ljungberg. 2011. "Open source in the firm: Opening up professional practices of software development." Research Policy (40:4), pp. 576-587.

Rolland, KH and Monteiro, E. 2007 "When 'perfect' integration leads to increasing risks: the case of an integrated information system in a global company", In: Risk, complexity and ICT, C. Ciborra and O. Hanseth (eds.), Edwar Elgar Publishing

Roquilly, C., 2010. The control over virtual worlds by game companies: issues and recommendations. MIS Quarterly (Forthcoming).

Rosenkopf, L, A Metiu, and VP George. 2001. "From the bottom up? Technical committee activity and alliance formation." Administrative Science Quarterly 46 (4) (December): 748-772.

Rosenkopf, L. Metiu A., V.P. George. 2001. From the bottom up? technical committee activity and alliance formation. Administrative Science Quarterly. 46 748-72.

Rothenberger, M.A., K.J. Dooley, U.R. Kulkarni, N. Nada. 2003. Strategies for Software Reuse: A Principal Compo-

nent Analysis of Reuse Practices. IEEE Transactions on Software Engineering, 29(9) 825-837.

Rouse, J. 2007. "Social practices and normativity." Philosophy of the social sciences (37:1), 46-56.

Rullani, F. 2007. "Dragging developers towards the core. How the Free/Libre/Open Source Software community enhances developers' contribution," In EURAM annual meeting.

Ryan, R.M., Deci, E.L., 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. American Psychologist. 55 68-78

Sabatier, V, V Mangematin, and T Rousselle. 2010. "From Recipe to Dinner: Business Model Portfolios in the European Biopharmaceutical Industry." Long Range Planning 43 (2-3) (June): 431-447.

Sampler, JL. 1998. "Redefining industry structure for the information age." Strategic Management Journal 19 (4) (April): 343-355.

Sarasvathy, S., 2004. Making it happen: beyond theories of the firm to theories of firm design. Entrepreneurship Theory and Practice 28 (6), 519–531.

Sarasvathy, S., N. Dew. 2005. New market creation through transformation. Journal of Evolutionary Economics 15(5) 533-565.

Sarasvathy, S.D., 2001. Causation and effectuation: toward a theoretical shift from economic inevitability to entrepreneurial contingency. The Academy of Management Review 26 (2), 243–263.

Sawhney, M., E. Prandelli. 2000. Communities of creation: Managing distributed innovation in turbulent markets. California Management Review 42(4) 24.

Scacchi, W. 2002. "Understanding Requirements for Developing Open Source Software Systems," IEEE Proceedings - Software (149:1), pp. 24-39.

Schatzki, T. R. 2005. "Peripheral Vision: The Sites of Organizations," Organization Studies (26:3), pp. 465-484.

Schatzki, T. R., Knorr-Cetina, K. and Savigny, E. V. 2001. The Practice Turn in Contemporary Theory, (1st ed,)London, UK: Routledge.

Schau, HJ, and MC Gilly. 2003. "We are what we post? Self-presentation in personal Web space." Journal of Consumer Research 30 (3) (December): 385-404.

Schmidt, D.C., M. Fayad, R.E. Johnson. 1996. Introduction. Communications of the ACM. 39(10) 36-39.

Schoberth, T., Heinzl, A., & Preece, J. 2006. Exploring communication activities in online communities: A longitudinal analysis in the financial services industry. Journal of Organizational Computing and Electronic Commerce, 16(3-4): 247-265.

Schofield, A. and Cooper, G.S., 2006. "Participation in Free and Open Source Communities: An Empirical Study of Community Members' Perceptions." In: Open Source Systems. Springer, pp. 221-231.

Schumpeter, J.A., 1934. The Theory of Economic Development. Harvard University Press, Cambridge.

Scott, A.J., 2004. Hollywood and the world: the geography of motion-picture distribution and marketing. Review of International Political Economy 11 (1), 33–61.

Sein, M.K., O. Henfridsson, S. Purao, M. Rossi, R. Lindgren, 2011. Action Design Research. MIS Quarterly. Forthcoming.

Selten, R. 1967. Die Strategiemethode zur Erforschung des eingeschränkt rationalen Verhaltens im Rahmen eines Oligopolexperimentes. Sauermann, H. (ed). Beiträge zur experimentellen Wirtschaftsforschung (pp. 136-168). Tübingen: J.C.B. Mohr (Paul Siebeck).

Setia, P., B. Rajagopalan, V. Sambamurthy, R. Calantone. 2010. How Peripheral Developers Contribute to Open-Source Software Development. Information Systems Research.

Shah, S., Tripsas, M., 2007. The accidental entrepreneur: the emergent and collective process of user entrepreneurship. Strategic Entrepreneurship Journal 1, 123–140.

Shah, S.K. 2006. Motivation, Governance and the Viability of Hybrid Forms in Open Source Software Development. Management Science. 52 1000-1014.

Shane, S., Venkataraman, 2000. The promise of entrepreneurship as a field of research. Academy of Management Review 25 (1), 217–226.

Shankar, V., Smith, A. K., & Rangaswamy, A. 2003. Customer satisfaction and loyalty in online and offline environments. International Journal of Research in Marketing, 20(2): 153-175.

Sheldon, K. M., and Krieger, L. S. 2007. "Understanding the negative effects of legal education on law students: A longitudinal test of self-determination theory," Personality and Social Psychology Bulletin (33:6), pp. 883.

Shirky, Clay. 2005. A group is its own worst enemy. In The Best Software Writing I: Selected and introduced by Joel Spolsky. Apress. http://www.springerlink.com/content/x768266j5m1l2474/ .

Siggelkow, N., 2007. Persuasion with case studies. Academy of Management Journal 50 (1), 20–24.

Simon, H.A. 1996. The sciences of the artificial. MIT Press.

Smith, V. 1982. Microeconomic systems as experimental science. American Economic Review 72 923-955.

Snow, C.C., O.D. Fjeldstad, C. Lettl, R.E. Miles. 2010. Organizing Continuous Product Development and Commercialization: The Collaborative Community of Firms Model. The Journal of Product Innovation Management 28(1) p.3-16.

Sosa, M.E. 2011. Where Do Creative Interactions Come From? The Role of Tie Content and Social Networks. Organization Science 22(1) p.1-21.

Sosa, M.E., Eppinger S.D., C.M. Rowles. 2004. The misalignment of product architecture and organizational structure in complex product development. Management Science. 50(12) 1674-1689.

Spaeth, S, M Stuermer, and G von Krogh. 2010. "Enabling knowledge creation through outsiders: towards a push model

of open innovation." International Journal of Technology Management 52 (3-4): 411-431.

Spaeth, S., Haefliger, S., von Krogh, G. and Renzl, B., 2008. Communal resources in open source software development. Information research 13: available online http://informationr.net/ir/13-1/paper332.html.

Spender, J.-C., 2006. Getting value from knowledge management. TQM Magazine 18 (3): 238-254.

Spinosa, C., Flores, F., & Dreyfus, H. L. 1997. Disclosing new worlds. MIT Press.

Sproull, L., Dutton, W., Kiesler, S. 2007. Introduction to the special issue: Online communities. Organization Studies. 28(3) 277-281.

Sproull, Lee, and Sara Kiesler. 1986. "Reducing Social Context Cues: Electronic Mail in Organizational Communications." Management Science 32 (11) (November 1): 1492-1512.

Stake, R.E. 1995. The art of case study research. Sage.

Stallman, R. 1999. "The GNU Operating System and the Free Software Movement," In Open Sources: Voices from the Open Source Revolution, C. DiBona, S. Ockman, and M. Stone (eds.), O'Reilly & Associates, Inc., pp. 53-70.

Stallman, R.M. 2004. Free but shackled – the Java Trap. URL: http://www.gnu.org/philosophy/java-trap.html

Stam, W. 2009. "When does community participation enhance the performance of open source software companies?" Research Policy 38 (8) (October): 1288-1299.

Staudenmayer, N. Tripsas, M., C.L. Tucci. 2005. Interfirm modularity and its implications for product development. Journal of Product Innovation Management. 22(4) 303-321.

Steiniger, S., E. Bocher. 2009. An overview on free and OS GIS developments. International Journal of Geographical Information Science 23(10) 1345-1370.

Steinke, I.A., 2004. Quality criteria in qualitative research. In: Flick, U., von Kardoff, E., Steinke, I.A. (Eds), Companion to Qualitative Research., 184-190, Sage, London.

Stephan, P. 1996. The Economics of Science. Journal of Economic Literature 34 1199-1235.

Stewart, K. J., Ammeter, A. P., and Maruping, L. M. 2006. "Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects," Information Systems Research (17:2), pp. 126-144.

Stewart, K. J., and Gosain, S. 2006. "The Impact of Ideology on Effectiveness in Open Source Software Development Teams," MIS Quarterly (30:2), pp. 291-314.

Steyaert, C. 2007. 'Entrepreneuring' as a conceptual attractor? A review of process theories in 20 years of entrepreneurship studies. Entrepreneurship and Regional Development 19(6) 453-477.

Strandburg, K.J., 2008. Users as innovators: implications for patent doctrine. University of Colorado Law Review, 79 (46): 467–544.

Strauss, A.L., Corbin, J., 1990. Grounded theory research: procedures, canons and evaluative criteria. Zeitschrift für Soziologie 19 (6), 418 ff.

Stuermer, M., S. Spaeth, G. von Krogh. 2009. Extending private-collective innovation: a case study. R & D Management 39(2) 170-191.

Suchman, L., Blomberg, J., Orr, J., and Trigg, R. 1999. "Reconstructing technologies as social practice," American Behavioral Scientist (43:3), pp. 392–408.

Susman, G., R. Evered. 1978. Assessment of scientific merits of action research. Administrative Science Quarterly 23(4) 582-603.

Szulanski, G., 2000. The process of knowledge transfer: A diachronic analysis of stickiness. Organizational Behavior and Human Decision Processes 82 (1): 9-27.

Takeishi, A. 2002. Knowledge partitioning in the interfirm division of labor: The case of automotive product development. Organization Science 13 321-338.

Teece, D.J. 1977. Technology transfer by multinational firms: The resource cost of transferring technological know-how. The Economic Journal 87 242-261.

Teece, D.J. 1986. The Multinational Corporation and the Resource Cost of International Technology Transfer. Cambridge: Ballinger.

Teece, D.J., 1986. Profiting from technological innovation: implications for integration, collaboration, licensing and public policy. Research Policy 15, 285–305.

Thompson, M. 2005. "Structural and epistemic parameters in communities of practice." Organization Science 16 (2) (April): 151-164.

Tilly, Charles (1978). From mobilization to revolution. Reading, MA: Addison-Wesley.

Tippins, MJ, and RS Sohi. 2003. "ICT competency and firm performance: Is organizational learning a missing link?" Strategic Management Journal 24 (8) (August): 745-761.

Torvalds, L., and Diamond, D. 2001. Just for Fun, Texere.

Tracz, W. 1995. Confessions of a used program salesman: Instiutionalizing software reuse. Addison-Wesley.

Ulhøi, J. P. 2004. "Open source development: a hybrid in innovation and management theory," Management Decision (42:9), pp. 1095-1114.

Ulrich, K. 1995. The Role of Product Architecture in the Manufacturing Firm. Research Policy. 24 419-440.

Useem, B. (1998). Breakdown theories of collective action. Annual Review of Sociology, 24, 215-238.

Utterback, J.M. 1994. Mastering the dynamics of innovation. Harvard Business School Press.

van Aken, J. 2004. Management research based on the paradigm of the design sciences: The quest for field-tested and grounded technological rules. Journal of Management Studies 41(2) 219-246.

van der Burg, S., and van Gorp, A. 2005. "Understanding moral responsibility in the design of trailers," Science and Engineering Ethics (11:2), pp. 235-256.

van Maanen, J. 1979. The fact of fiction in organizational ethnography. Administrative Science Quarterly. 24 539-549.

Varela, F., E. Thompson, E. Rosch. 1991. The Embodied Mind: Cognitive Science and Human Experience. MIT Press.

Vaughan-Nichols, S. 2009. "If IBM owns Java ..." JavaWorld. April 3. http://www.javaworld.com/javaworld/jw-04-2009/jw-04-if-ibm-owns-java.html.

Vaughan, D., 1990. Autonomy, interdependence, and social control: NASA and the space shuttle challenger. Administrative Science Quarterly 35 (2), 225–257.

Vaughn-Nichols, S. J. 2009. "Linux Runs the Stock Exchange," PCWorld Business Center. Sept. 6. URL: http://www.pcworld.com/businesscenter/article/171523/linux_runs_the_stock_exchange.html.

Venkataraman, S., 1997. The distinctive domain of entrepreneurship research. In: Katz, J. (Ed.), Advances in Entrepreneurship, Firm Emergence, and Growth, vol. 3. JAI Press, Greenwich, CT, pp. 119–138.

von Hippel, E. 1987. Cooperation Between Rivals: Informal Know-How Trading. Research Policy. 16 291-302.

von Hippel, E. 2001. Learning from open-source software. MIT Sloan Management Review 42(4) 82–86.

von Hippel, E. 2001. User toolkits for innovation. Journal of Product Innovation Management. 18(4) 247 257.

von Hippel, E. 2005. Democratizing Innovation. MIT Press. URL: http://web.mit.edu/evhippel/www/democ.htm

von Hippel, E., 1987. Cooperation between rivals: informal know-how trading. Research Policy 16, 291–302.

von Hippel, E., 1988. The Sources of Innovation. Oxford University Press, Oxford.

von Hippel, E., 1994. Sticky information and the locus of problem solving: implications for innovation. Management Science 40 (4), 429–439.

von Hippel, E., 1998. Economics of product development by users: The impact of "sticky" local information. Management Science 44 (5): 629-644.

von Hippel, E., 2005. Democratizing Innovation. MIT Press, Cambridge, MA.

von Hippel, E., 2007. Horizontal innovation networks—by and for users. Industrial and Corporate Change 16 (2), 293–315.

von Hippel, E., G. von Krogh. 2003. The Private-Collective Innovation Model in Open Source Software Development: Issues for organization science. Organization Science. 14(2) 209-223.

von Hippel, E., G. von Krogh. 2006. Free revealing and the private-collective model of innovation incentives. R&D Management 36 295-306.

von Hippel, E., K.R. Lakhani. 2003. How open source software works: "free" user-to-user assistance. Research Policy. 32(6) 923-943.

von Krogh, G, E. von Hippel. 2003. Special issue on open source software development. Research Policy 32(7) p.1149-1157.

von Krogh, G. 2002. "The communal resource and information systems." Journal of Strategic Information Systems 11 (2) (June): 85-107.

von Krogh, G. K. Ichijo, I. Nonaka. 2000. Enabling knowledge creation: Unlocking the mystery of tacit knowledge and unleashing the power of innovation. Oxford University Press.

von Krogh, G., 1998. Care in knowledge creation. California Management Review 40 (3): 133-153.

von Krogh, G., 2003. Knowledge sharing and the communal resource. In: M. Easterby-Smith and M.A. Lyles (Eds.), Handbook of Organizational Learning and Knowledge Management., 372-392, Blackwell, Malden, MA.

von Krogh, G., E. von Hippel. 2003. Special issue on open source software development. Research Policy 32(7) p.1149-1157.

von Krogh, G., E., von Hippel. 2006. The promise of research on Open Source software. Management Science. 52(7) 975-983.

von Krogh, G., Ichijo, K. and Nonaka, I., 2000. Enabling knowledge creation: Unlocking the mystery of tacit knowledge and unleashing the power of innovation. Oxford University Press, Oxford.

von Krogh, G., S. Haefliger, 2010. "Opening up design science: The challenge of designing for reuse and joint development." Journal of Strategic Information Systems 19: 232-241.

von Krogh, G., S. Spaeth, K.R. Lakhani. 2003. Community, joining, and specialization in open source software innovation: A case study. Research Policy. 32(7) 1217-1241.

von Krogh, Georg, C. Rossi Lamastra, S. Haefliger. 2012. Phenomenon-based research in management and organization science: When is it rigorous and does it matter? Long Range Planning Forthcoming.

Wagner, EL, S Newell, and G Piccoli. 2010. "Understanding Project Survival in an ES Environment: A Sociomaterial Practice Perspective." Journal of the Association for Information Systems 11 (5): 276-297.

Wasko, J., Phillips, M., Purdie, C., 1993. Hollywood meets Madison Avenue: the commercialization of US films. Media, Culture & Society 15, 271–293.

Wasko, M.M., S. Faraj 2000. "It is what one does": why people participate and help others in electronic communities of practice. Journal of Strategic Information Systems 9 155-173.

Wasko, M.M., S. Faraj. 2005. Why Should I Share? Examining Social Capital and Knowledge Contribution in Electronic Networks of Practice. MIS Quarterly 29(1) p.35-37.

Webb, C., 2009. Dear diary: Recommendations for researching knowledge transfer of the complex. Electronic Journal of Knowledge Management 7 (1): 191-198.

Weber, K., M.T. Dacin. 2011. The Cultural Construction of Organizational Life: Introduction to the Special Issue. Organization Science 22 p.287–298.

Weber, R. 2003. Learning with no feedback in a competitive guessing game. Games and Economic Behavior 44 134-144.

Weber, R., C. Camerer. 2003. Cultural Conflict and Merger Failure: An Experimental Approach. Management Science 49 400-415.

Weber, R., Y. Rottenstreich, C. Camerer , M. Knez. 2001. The illusion of leadership: Misattribution of cause in coordination games. Organization Science 12 582-598.

Weick, K.E. 1976. Educational organizations as loosely coupled systems. Administrative Science Quarterly. 21(1) 1-19.

Wenger, E. and Snyder, W.M., 2000. Communities of practice: the organizational frontier. Harvard Business Review 1: 139-145.

Wenger, E., 1999. Learning as social participation. Knowledge Management Review 6: 30-33.

Wenger, E., 2000. Communities of practice and social learning systems. Organization 7 (2): 225 -246.

Wenger, E., McDermott, R. and Snyder, W.M., 2002. Cultivating communities of practice: A guide to managing knowledge. Harvard Business School Press, Boston, M. A.

West, J. 2003. How open is open enough? Melding proprietary and open source platform strategies. Research Policy 32(7) 1259.

West, J., and O'Mahony, S. C. 2005. "Contrasting Community Building in Sponsored and Community Founded Open Source Projects," In Proceedings of the 38th Annual Hawaii International conference on System Sciences (Jan 2005).

West, Joel, and Scott Gallagher. 2006. "Challenges of open-innovation: the paradox of firm investment in open-source software." R&D Management 36 (3) (June 1): 319-331.

Whitley, E.A., L. Willcocks. 2011. Achieving Step-Change in Outsourcing Maturity: Toward Collaborative Innovation. Mis Quarterly Executive 10(3) p.95-107.

Whittington, R. 2006. "Completing the practice turn in strategy research," Studies (27:5), pp. 613–634.

Wielaard, M. 2006. Toward a free Java. LWN.net. URL: http://lwn.net/Articles/184967

Wiertz, C., K. de Ruyter. 2007. Beyond the call of duty: Why customers contribute to firm-hosted commercial online communities. Organization Studies 28(3) p.347-376.

Wirtz, BW, O Schilke, and S Ullrich. 2010. "Strategic Development of Business Models Implications of the Web 2.0 for Creating Value on the Internet." Long Range Planning 43 (2-3) (June): 272-290.

Wolf, P. and Wunram, M., 2003. Barriers to KM between organisational cultures in the face of concurrent enterprising: How to overcome them?. In: Camarinha-Matos, L.M. and Afsarmanesh, H. (Eds.), Processes and Foundations for Virtual Organisations, 333-340, Kluwer Academic Publishers, Boston.

Wu, C., Gerlach, J. H., and Young, C. E. 2007. "An Empirical Analysis of Open Source Software Developers' Motivations and Continuance Intentions," Information & Management (44:3), pp. 253-262.

Xu, B., Jones, D. R., and Shao, B. 2009. "Volunteers' involvement in online community based software development," Information & Management (46:3), pp. 151-158.

Yalta, A.T., A.Y. Yalta. 2010. Should Economists Use OS Software for Doing Research? Computational Economics 35(4) 371-394.

Yamauchi, Y., M. Yokozawa, T. Shinohara, T. Ishida. 2000. Collaboration with Lean Media: How Open-Source Software Succeeds. ACM Conference on Computer Supported Cooperative Work. Philadelphia, PA. URL: http://www.lab7.kuis.kyoto-u.ac.jp/~ishida/pdf/cscw00.pdf

Yang, J.M., Li, K.F., Zhang, D.F. 2009. Recommendation based on rational inferences in collaborative filtering. Knowledge-Based Systems, 22(1): 105-114.

Ye, Y., and Kishida, K. 2003. "Toward an Understanding of the Motivation of Open Source Software Developers," In Proceedings of 2003 International Conference on Software Engineering (ICSE2003), pp. 419-429.

Yeow, A.Y.K., C.E.H. Chua. 2010. Artifacts, Actors, and Interactions in the Cross-Project Coordination Practices of Open-Source Communities. Journal of the Association for Information Systems 11(12).

Yin, R.K., 2003. Case study research: design and methods. Sage, Thousand Oaks, CA.

Yoo, Y., O. Henfridsson, and K. Lyytinen. 2010. "The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research." Information Systems Research 21 (4): 724-735.

Yoon, H., Malecki, E,J., 2009. Cartoon planet: worlds of production and global production networks in the animation industry. Industrial and Corporate Change 19 (1), 239–271.

Young, R., W.G. Rohm. 1999. Under the Radar: How Red Hat Changed the Software Business - and Took Microsoft by Surprise. Coriolis Group Books.

Yu, J., Jiang, Z., and Chan, H. C. 2007. "Knowledge Contribution in Problem Solving Virtual Communities: the Mediating Role of Individual Motivations," In Proceedings of the 2007 ACM SIGMIS CPR 2007 conference on computer personnel doctoral consortium and research conference, pp. 144-152.

Zander, U., B. Kogut. 1995. Knowledge and the speed of the transfer and imitation of organizational capability: An empirical test. Organization Science. 6(1) 76-92.

Zeitlyn, D. 2003. "Gift economies in the development of open source software: anthropological reflections," Research Policy (32:7), pp. 1287-1291.

Zhang, J. & Baden-Fuller, Charles, 2010. The Influence of Technological Knowledge Base and Organizational Structure on Technology Collaboration. Journal of Management Studies, 47(4), pp.679-704.

Zhu, K., K.L. Kraemer, V. Gurbaxani, S.X. Xu. 2006. Migration to open-standard interorganizational systems: Network effects, switching costs, and path dependency. MIS Quarterly 30 p.515-539.

Zwick, R. X.P. Chen. 1999. What Price Fairness? A Bargaining Study. Management Science 45 804-823.